UNIVERSIDAD CARLOS III DE MADRID.
ESCUELA POLITÉCNICA SUPERIOR DE LEGANÉS

GRADO EN INGENIERÍA EN TECNLOGÍAS DE TELECOMUNICACIÓN

TRABAJO DE FIN DE GRADO

ALMACENAMIENTO Y SINCRONIZACIÓN DE FICHEROS ALTERNATIVO A LA NUBE MEDIANTE P2P E INTERFAZ WEB

Guillermo Fernández Gorostidi
Tutor: Daniel Díaz Sánchez

Junio 2016

Agradecimientos

Quiero darle las gracias a Dani, que ha tenido una paciencia infinita a lo largo de este proyecto, tanto para explicarme las cosas varias veces si hacía falta, como para charlar un rato en su oficina sobre como iban las cosas o darme consejos que no tenían que ver con el TFG.

También gracias a mis padres que tan comprensivos son conmigo y me apoyan siempre, aunque a veces diga cosas que sean difíciles de entender o de aceptar. Está por llegar el día en que no me deis todo lo que necesito y mucho más. Os quiero muchísimo.

Y a mis amigos, a los que os llevo tanto tiempo hablando de este proyecto que por fin se ha materializado. Ojalá poder aportaros a vosotros lo que vosotros me habéis aportado a mí.

De verdad, gracias a todos.

Resumen

Los servicios online suelen ofrecerse siguiendo el modelo clásico de cliente-servidor, donde nos comunicamos en nuestro rol de cliente con un servidor que provee el servicio. Sin embargo, a medida que crece Internet se populariza el modelo Peer-to-Peer. En este tipo de redes los recursos son compartidos entre todos los nodos de la red, aumentando considerablemente la capacidad disponible para cada uno. En estas redes no hay clientes y servidores, si no que cada nodo es ambas cosas a la vez.

Un servicio que sigue el modelo cliente-servidor al que estamos muy acostumbrados es La Nube: un número desconocido de servidores en localizaciones desconocidas que nos permiten utilizar sus recursos. Especialmente para almacenamiento de archivos, estos servicios se han popularizado enormemente en los últimos años, principalmente por la comodidad de tener nuestros archivos disponibles en cualquier sitio, en cualquier momento.

A pesar de las ventajas de este tipo de servicios, subir archivos a un servidor del que no tenemos el más mínimo control tiene muchas desventajas. Aunque no nos demos cuenta, estamos subiendo nuestros archivos personales a un servidor al que no sabemos quién tiene acceso, dónde esta localizado o cómo guarda exactamente nuestros archivos.

En este proyecto se diseña una solución a estos problemas de seguridad y privacidad mediante una red P2P. Más concretamente, el modelo utilizado es el de una red Friend-to-Friend, un tipo concreto de red P2P en la que solo determinados nodos de confianza pueden interactuar entre ellos. Esto resultará en una suerte de carpeta compartida en la que solamente los usuarios que formen parte de la red podrán almacenar e intercambiar archivos a través de ella. Para ofrecer todas las funcionalidades que ofrecería un servicio de almacenamiento basado en La Nube, habrá un Interfaz Web que será capaz de conectarse al resto de componentes de la subred, permitiendo así el acceso a los archivos desde cualquier dispositivo de Internet.

Se hará una prueba de concepto en la que se probará que esta solución es implementable con una estructura modular de los nodos y sin necesidad de servicios de terceros. Todo ello se construirá sobre una red lógica superpuesta a la red IP siguiendo el modelo P2P.

Índice General

Índice de Figuras

Capítulo 1

1. Introduction

1.1 Project Motivation

The networks to which we refer as traditional, the ones that are based in a client-server structure, are the most used ones nowadays to provide the majority of Internet's services. These traditional networks have their advantages and disadvantages, but are slowly giving way to Peer-to-Peer networks.

This migration phenomenon towards P2P is one that we can already be seen in several services. The reason for this to be happening is that the performance of these networks in which resources are shared is higher. The performance increases when the number of nodes interconnected increases; scalability of P2P networks make of its use an unavoidable milestone in the history of the Internet. In this way, there is always a P2P version of any service that can be provided with the traditional structure, and the ones that haven't come out yet are gradually being developed and deployed.

In the case of this project, the idea is based on the storage and synchronization of files. Storage in The Cloud is something that we are not only already used to, but rather on which we count for a great deal of our online activities throughout our day. Whether it is to make a backup copy of our files, in order to have them available at work or university, or with any other objective, more and more we tend to not only depend on the local resources of our devices.

In spite of all the advantages that these Cloud services offer to us, two of the most important aspects, if not the most important, are overlooked: the privacy and security of the files. When we upload files to the server of a third party, we hand in the complete control over these files; we no longer know where they are located, how they are stored or who can actually be granted access to them. What in local storage is basic and strictly under the user's control, in The Cloud can be compromised too easily.

It is because of this that the idea of developing a file synchronization application among several devices or users using a P2P network starts. There are already countless applications available on The Internet which grant access to P2P networks in order to download and share files with other users, even if we have never, or will never, know them. Nevertheless, this project is more centered on sharing and synchronizing files with known authenticated users, more in the Friend-to-Friend network fashion.

The security issue will be solved as much as the user has a secure local environment. That is, the files that are stored and managed by the application will be as secure as the user of the machine wants them to be; the control of the security is back to the hands of the user.

In the same way, all the files will travel encrypted from one user to another, and only the users that form the network will actually be able of reading and understanding the files that are shared. If an attacker would intercept in some intermediate router any of the packets belonging to a file, he or she would not be able to know what is being shared.

There exist some initial versions of similar programs published on The Internet but they are not as popular as one would expect, given the advantages they have over Cloud storage. The reason for this is that, more often than not, the user prefers the convenience of being able to access their files from any device and not only from those which take part in the shared folder. This is achieved in Cloud systems by means of connecting to the storage server through the web.

This disadvantage will try to be overcome in this project by developing a web that will communicate directly with the devices that form the shared folder. In this way, the user will have access to the shared files from any device as long as it has Internet connection.

1.2 Objectives

This project will have as its main objective to create a file sharing and synchronization service over an overlay F2F network that is also accessible through a web interface that connects directly with the conforming nodes. For this purpose, the implementation of the following elements will be needed:

- A node structure will be created that will have identical functionalities. They will be able to communicate with each other and to share information and files in a private way through a F2F network superposed to the physical one.

- The file sharing will happen based on some information stored in structures designed specifically for this project. These structures will be present in every node and they will store information about the availability of each part of each file in that node.

- Also, the nodes will need to use a communication protocol to respond to all the possible actions that can take place in the network.

- As this communication protocol will also have to be used by the web interface that connects to the rest of the network, it will be defined as a REST API.

- It will be necessary to develop a web interface that is attractive to the user and that offers full operability with all the functionalities of the application.

- Finally, to deal with the security issues, an authentication method will be designed for the nodes that want to form part of the network, including the web interface node.

1.3 Contents of the Document

This document will comprise the following chapters.

- Chapter 1 – Introduction
  Contains an introduction to the topic of file synchronization in the Cloud and its alternative with F2F networks. It will have a short motivation of the project, an objective declaration and the present list of contents.

- Chapter 2 – State of the art
  Exposes the state of the art of P2P/F2F networks and gives some examples and explanations of already developed applications.

- Chapter 3 – Design
  Describes the process of design of a structured F2F network for file synchronization. More specifically, it will be explain in the modules that conform the node, the basic entity of the project.

- Chapter 4 – Development
  Explains the development of the different components of the application: the nodes and the communication between them, the structure of the network and the web interface access.

- Chapter 5 – Testing
  Contains the evaluation criteria of the main objectives, as well as the tests and analysis performed.

- Chapter 6 – Planning and budget
  Describes in detail the different stages of the project, based on time and the activities to be performed, as well as an estimated budget.

- Chapter 7 – Conclusions and future work
  Closes the project exposing the conclusions reached, and gives some possible ideas for future work on the application.

- Chapter 8 – Legal environment
  Analyses briefly the legal environment in which this kind of networks and applications are placed and how it affects the project.

Capítulo II

2. Estado del Arte

2.1 Redes Peer-to-Peer

2.1.1 Modelo Cliente-Servidor

Este modelo se basa en una diferenciación entre dos tipos de nodos, los clientes y los servidores, teniendo cada uno tendrá unas funciones específicas.

Los clientes realizarán peticiones a los servidores, ya sea de utilización de sus recursos o de descarga de un archivo particular que el servidor tenga. Los clientes no se comunicarán entre sí ni harán uso de los recursos del resto de los clientes.

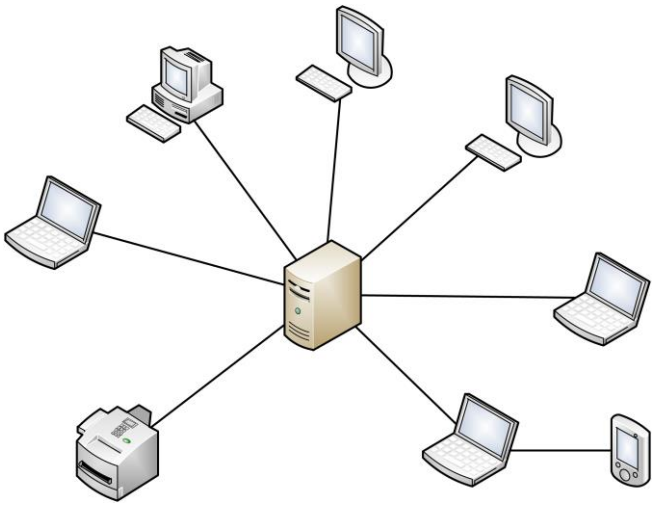


Figura 1: Modelo Cliente-Servidor

Por el contrario, los servidores responderán con sus servicios y recursos a las peticiones de los clientes. Normalmente serán máquinas con mucha más capacidad que los clientes a los que sirve, ya que tendrán que dar respuesta a un número alto de peticiones.

Aunque será una red más fácil de controlar que la Peer-to-Peer, será menos eficiente y mucho más cara a causa de la necesidad de un servidor potente. Además, en caso de escalarse el desempeño de la red se vería muy reducido y habría que hacer una nueva inversión en nuevos servidores.

2.1.2 Modelo Peer-to-Peer

Una red Peer-to-Peer es una estructura de nodos distribuidos en la que dichos nodos son idénticos en funcionalidades. Esto tendrá una importancia vital, ya que no utilizan el modelo tradicional anteriormente mencionado en el que un servicio es proporcionado a un cliente que lo solicita, por un servidor que se dedica a proporcionarlo.

En el caso de las redes Peer-to-Peer (a las que nos referiremos de ahora en adelante como redes P2P), todos los nodos juegan el papel tanto de servidor como de cliente. De esta manera, los recursos disponibles para cada nodo se multiplican, al poder sacarlos de un número mucho mayor de componentes de la red. Esto tendrá implicaciones en cuanto a la tolerancia a los fallos, la capacidad del sistema y la adaptabilidad del mismo [1].

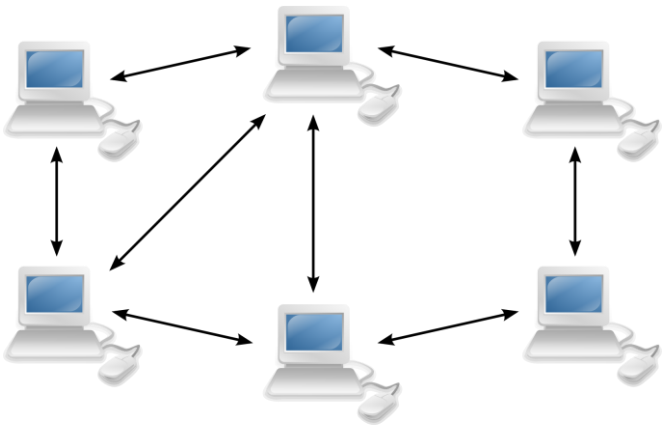


Figura 2: Red Peer-to-Peer

Una red P2P se construye sobre redes físicas de las que será un subconjunto sin una relación constante. La red lógica de nodos se

superpone a la red física, que normalmente está constituida por Internet. Así, aunque el tráfico fluya a través de la red IP, los nodos podrán comunicarse de manera directa a nivel de aplicación.

La utilización más eficiente de los recursos mediante el uso distribuido de los mismos, ya sean archivos, capacidad de procesamiento, etc., dota al sistema de una mayor capacidad. Procesos y trabajos que son excesivamente grandes o costosos para un nodo aislado, pasan a ser factibles al compartirse la carga entre todos los nodos. Así, a medida que aumenta el número de nodos, aumenta también la rapidez y la capacidad para realizar estos trabajos.

Con respecto a la tolerancia a los fallos, las redes P2P tienen también ciertas ventajas. Al estar los nodos conectados, pero no depender directamente unos de otros, si uno o varios de ellos falla, no compromete al resto de la red, que podrá seguir operando perdiendo solamente un diferencial de su capacidad.

Finalmente y muy en relación con las dos características anteriores, al estar la red descentralizada también lo estará la información sobre dicha red. Es decir, conocer la estructura completa de la red es innecesario, quizá incluso imposible si la red es suficientemente grande, y solamente hará falta obtener información de los nodos adyacentes (que no tendrán por qué coincidir con los más cercanos en la topología física) para poder operar con ella de manera eficiente.

2.1.3 Arquitectura

Hay principalmente tres maneras de clasificar las redes P2P según su estructura: centralizada, descentralizada y entre medias una híbrida denominada semicentralizada [1].

Centralizada

En este modelo existirá un nodo central que dará respuesta a todos los demás. Se diferencia de una red servidor-cliente tradicional en que no proveerá a los nodos de servicios, si no de información acerca de los demás nodos de la red. En el momento en que un nodo A quiera comunicarse con otro nodo B, le solicitará al nodo central la información necesaria que éste guarda en su base de datos. Una vez el nodo central le haya proporcionado dicha información, el nodo A podrá comunicarse con el nodo B sin necesidad de dirigir su tráfico a través del nodo central.

Las redes P2P de primera generación como Napster [1] se basaban precisamente en esta estructura, que fue desplazada a medida que éstas

redes avanzaban hacia modelos más descentralizados. La principal razón para dicha evolución fueron los puntos únicos de fallo que representaban los nodos centrales así como el cuello de botella que podían llegar a formar.

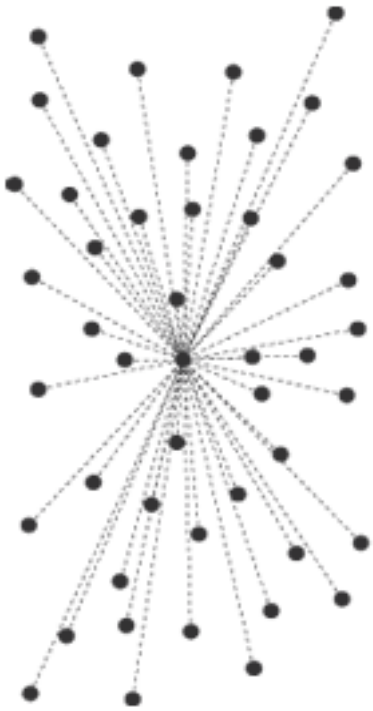


Figura 3: Red centralizada

Desde el punto de vista de la seguridad, si los nodos centrales dejaran de funcionar, la red dejaría de hacerlo con ellos, y de cara a la privacidad, con un ataque fructífero al nodo central se podría obtener toda la información necesaria acerca de la localización de los nodos de la red.

Descentralizada

En el modelo descentralizado no existe ningún nodo central, todas las comunicaciones se hacen de usuario a usuario, incluyendo las de control de la red. Es la opción que mejor tolera los problemas al no tener puntos únicos de fallo; la desaparición de cualquiera de los nodos no compromete la integridad de la red.

Es la red puramente P2P como se habría descrito al inicio de este capítulo y a pesar de que algunas redes actuales o recientes se basaban en este modelo, es difícil de conseguir ya que consume una cantidad de recursos relativamente grande solamente para el control y la búsqueda de otros nodos.

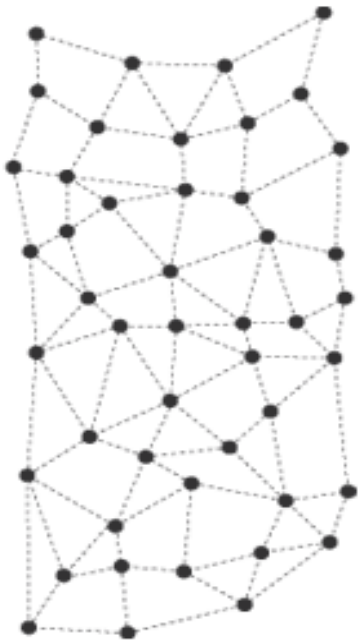


Figura 4: Red puramente distribuida

Híbrida

Entremedias de la red puramente centralizada y la red totalmente descentralizada se encuentra la red híbrida. Este tipo de red utilizará lo mejor de cada una, al haber varios nodos centrales que se dedicarán a dar servicio de enrutamiento y administración de recursos. Sin embargo, estos nodos centrales no serán nodos normales ya que no compartirán sus recursos propios con los demás.

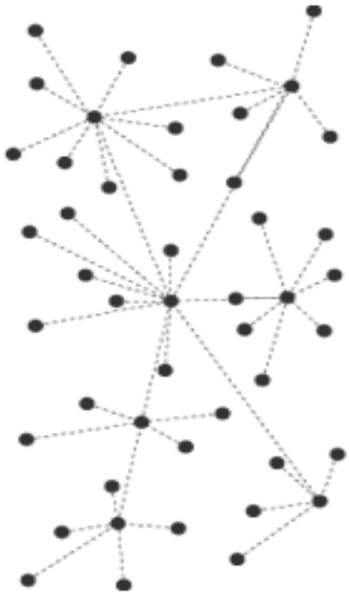


Figura 5: Red híbrida o semidescentralizada

Este enfoque tiene claras ventajas, por ejemplo en cuanto a la seguridad. Incluso aunque uno de los nodos centrales dejara de estar disponible, los otros podrían absorber su carga. Además, se optimiza el tráfico al conectarse los nodos centrales entre ellos y tener una visión mucho más completa de la arquitectura de la red.

En las generaciones posteriores se empieza a dar más esta arquitectura hasta llegar a ser a día de hoy la más utilizada. Un ejemplo de ello es el protocolo BitTorrent, uno de los más utilizados en Internet y que ayudó a inspirar este Trabajo de Fin de Grado.

2.1.4 Estructura

A diferencia de la clasificación de la arquitectura, en la que encontramos varios tipos diferenciados, en el caso de la estructura es más una cuestión de estructura o ausencia de ella. La estructura será importante en lo referente al encaminamiento de los paquetes y las conexiones entre los nodos, independientemente de la arquitectura [1].

Red desestructurada

En estas redes las conexiones entre los nodos se dan de manera indeterminada. Es decir, no hay una estructura previamente concretada de las conexiones entre los nodos de la red.

Si un nodo quiere un determinado recurso, al no saber quién puede proporcionárselo, utilizará un método denominado *flooding* o *inundación* por el que mandará mensajes pidiendo el recurso a todos los nodos adyacentes. Si estos no lo tuvieran, reenviarían el mensaje a todos sus nodos adyacentes, etc.

Ésta técnica puede llegar a generar una cantidad ingente de tráfico, ya que los mensajes se multiplican de manera exponencial, y además no garantiza que se encuentre el recurso pedido.

Redes con estructura

Por el contrario, en las redes con estructura la búsqueda es determinista ya que las conexiones entre nodos no se dan de manera arbitraria. Cada nodo tiene su propia tabla de encaminamiento, que se

implementa mediante *hash tables*, y se asignan identificadores únicos a cada archivo y cada nodo.

Los nodos no tendrán en su lista todos los identificadores, si no que cada uno se ocupará de un número concreto de ellos. La inmensa mayoría de las redes P2P actuales se sirven de estas *Distributed Hash Tables (DHT)* para estructurarse.

Así, cuando un nodo quiere un recurso concreto, sabe a quién pedírselo incluso aunque no sepa exactamente dónde se encuentra. Es también importante notar que cada vez que un archivo o nodo se introduce o retira de la red se deben actualizar las tablas, pero incluso esto se convierte en un proceso eficiente ya que sólo un cierto número de las tablas deben actualizarse con cada cambio.

2.1.5 Redes Overlay con Estructura

Como ya se ha tratado en la descripción inicial de las redes P2P, se estructuran de manera superpuesta sobre la red IP; a esto se le denomina redes overlay.

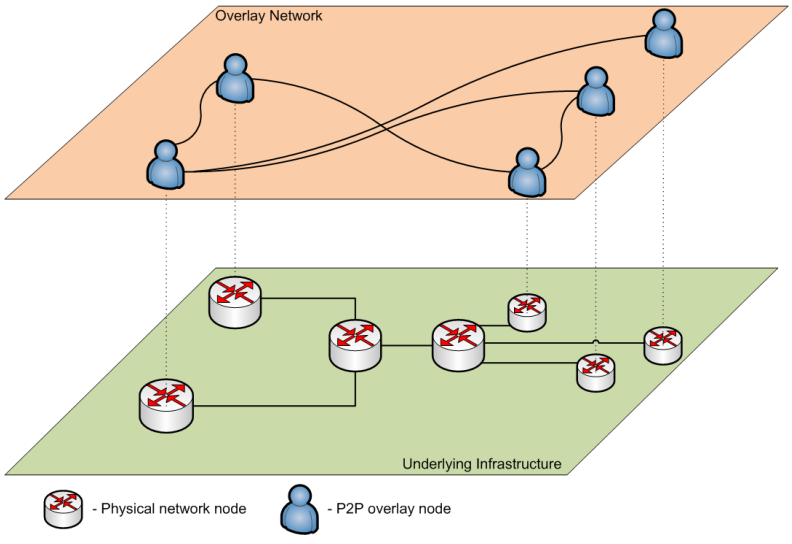


Figura 6: Red overlay superpuesta a la red física

La independencia que tienen las redes overlay y la eficiencia de las redes estructuradas ha inspirado varios protocolos y sistemas que se dan con frecuencia en las redes P2P actuales.

Diseñada en 2002 por Petar Maymounkov and David Mazières [2], es el ejemplo básico de una red overlay estructurada mediante una DHT. Cada nodo tendrá un identificador único de 160 bits que se construirá mediante una función hash de la dirección IP y se tratara como la hoja de un árbol binario.

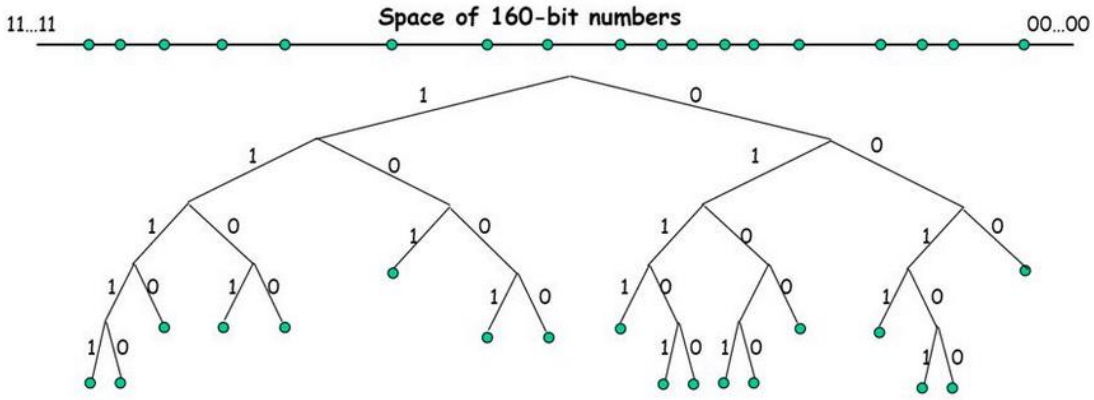


Figura 7: Árbol binario de identificadores

Estos identificadores permitirán calcular a qué distancia está un nodo de otro. Esto se calcula mediante una operación XOR ⊕ entre los identificadores del nodo de origen y el de destino. Es una manera rápida y sencilla de formar una tabla de rutas en base a la distancia a la que estén los nodos, siendo los más cercanos los que más arriba estén en la tabla, y los más lejanos los que estén al final de la tabla.

De esta manera cada nodo tendrá una distancia de cero consigo mismo:

$$distancia(A, A) = A \oplus A = 0$$

La distancia irá aumentando a medida que difiera el número de bits del identificador del nodo origen con respecto del identificador del nodo de destino. Además, debido a la simetría de la operación XOR ⊕, las distancias serán también simétricas entre dos nodos, sea el que sea el que se considere el origen y el que se considere destino. De esta manera habrá tantos niveles como bits se hayan asignado al identificador de la red.

Kademlia llama a las tablas *"k-buckets"*, donde k es el número de nodos que se van a almacenar para cada nivel de distancia. Es decir, en el nivel de la lista que está a distancia 1 habrá como máximo *k* nodos registrados, igual que para el nivel de la lista que representa la distancia N, donde N sea la distancia máxima. N es el número de bits del identificador, que en el caso de Kademlia es de 160 bits.

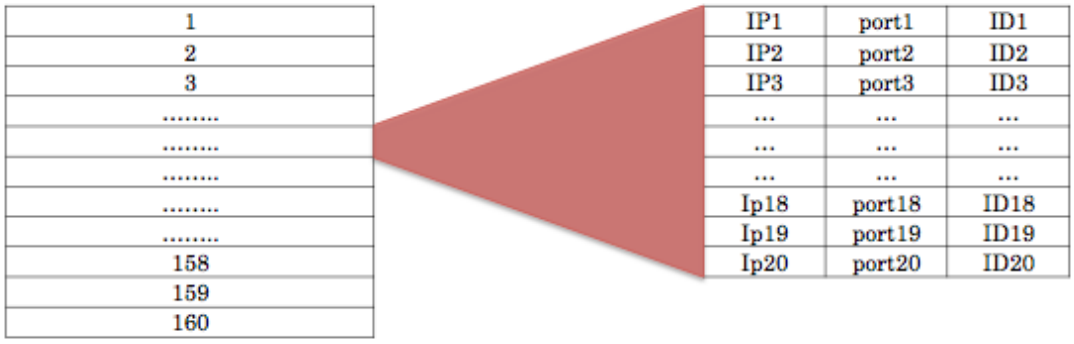


Figura 8: Tabla de distancias con k-buckets de k=20

Al ser más fácil encontrar nodos cercanos que lejanos, los primeros niveles se llenarán con más facilidad. Si un nuevo nodo tiene que ser registrado en un nivel que ya tiene *k* nodos registrados, se comprobará la conexión con cada uno de ellos y en caso de no responder alguno, será reemplazados por este nuevo registro. Si todos responden con normalidad el nuevo registro será guardado en una caché hasta que uno de los nodos registrados se considere desconectado y ocupe su lugar.

Para buscar un nodo o un recurso, se mandará un mensaje a los α nodos que más cerca estén del identificador de dicho nodo o recurso (típicamente $\alpha = 3$). Estos responderán con los identificadores de los nodos que sepan que están más cerca del recurso y el nodo origen actualizará sus tablas. Entonces mandará mensajes a estos nodos, que responderán con otros identificadores más cercanos aún, etc. Cuando los identificadores que se devuelven no están más cerca que los de las tablas del nodo origen, la búsqueda habrá finalizado.

Si el número de bits del identificador es 160, la búsqueda se hará en el peor de los casos en 160 iteraciones, ya que no buscará en el nivel 0 (correspondiente a sí mismo). Sin embargo el número de iteraciones suele ser mucho menor, dada la eficiencia del algoritmo y la dinamicidad de las tablas.

Este protocolo basado en Kademlia es a día de hoy uno de los más utilizados de la red. Su principal característica es la división de los archivos en pequeñas partes de igual tamaño, que podrán descargarse de diferentes nodos de manera independiente. Una vez se tengan todas las partes de un archivo se podrá re ensamblar para obtener el archivo completo. Este mecanismo ofrece sobretodo dos ventajas.

Desde el punto de vista de la eficiencia de la descarga, si tuviéramos que descargar un archivo completo de un solo nodo nos llevaría un determinado tiempo y ancho de banda. Esto es, si hubiera un único nodo del que descargar todo el archivo, una parte mucho mayor de su ancho de banda de subida estaría dedicado a enviar ese archivo, y durante más tiempo. En el caso de BitTorrent, el ancho de banda consumido por la descarga de una parte del archivo, así como el tiempo que se tarda en descargar dicho archivo, es mucho menor ya que se reparte la descarga entre varios nodos.

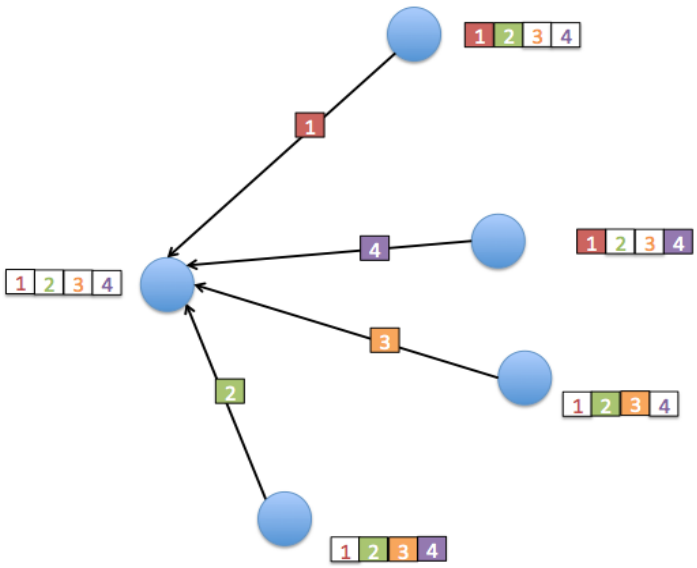


Figura 9: Descarga simultánea de diferentes partes del archivo

El otro punto ventajoso ocurre si se da el escenario en que una, o varias, partes de un archivo le llegan corruptas al nodo que descarga el archivo. Si no estuviera dividido en partes, el nodo tendría que volver a descargarse el archivo completo para sanearlo. Sin embargo, al saber

exactamente qué partes son las que están corruptas, el nodo puede volver a pedir solamente esas partes, ahorrando así tiempo y recursos.

Otro aspecto clave del protocolo BitTorrent es, a diferencia de la implementación pura de Kademlia, que no se forma una red puramente distribuida. Los nodos no saben quién tiene exactamente el archivo que ellos quieren, será el papel de los *trackers* proporcionarles esta información.

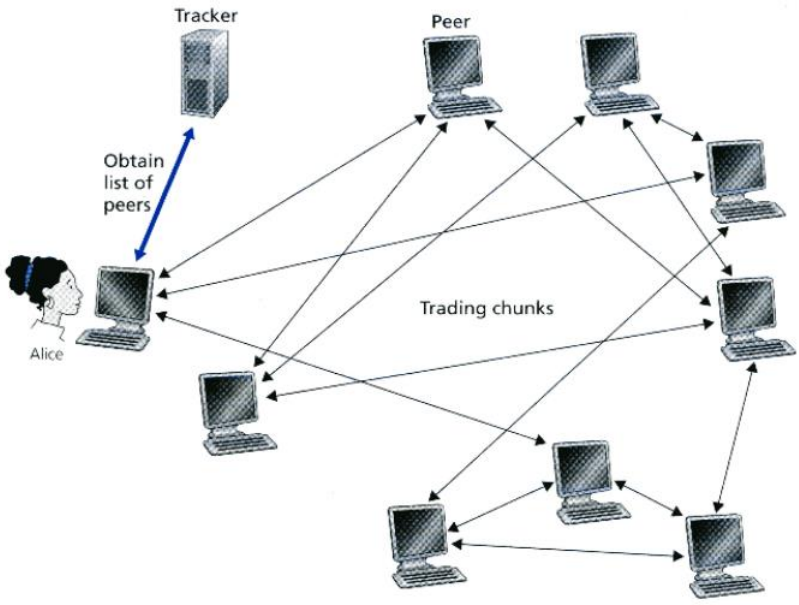


Figura 10: Funcionamiento del Tracker

La figura del *tracker* la llevará a cabo un servidor que tendrá la información sobre qué partes de un archivo tiene exactamente cada nodo. De esta manera, si un nodo quiere descargar un archivo le solicitará esta información al *tracker* y este le dirá a qué nodos puede pedir qué partes. A partir de ahí la comunicación ya será internodal sin que el *tracker* haga nada más que actualizar sus tablas.

TomP2P

Este proyecto de código abierto escrito en Java se inició en 2004 por Thomas Bocek como parte de su Tesis de Master [8]. A partir de esa primera versión, y viendo que la estructura tenía carencias a la hora de ser escalada, se hicieron varias versiones posteriores con comunicaciones

asíncronas utilizando objetos *future* y basadas en Java NIO para conseguir operaciones no bloqueantes.

Igual que BitTorrent, TomP2P se basa en Kademlia para implementar su DHT, guardando varios valores por cada clave. Estos valores podrán ser buscados y actualizados utilizando una clave secundaria.

Este framework condensa las características interesantes que se han tratado hasta ahora: es una red estructurada, utiliza una *Distributed Hash Table*; es puramente descentralizada, no habrá ningún servidor intermedio necesario para guardar la información acerca de los nodos; forma una red overlay, superponiendo una red lógica independiente de la red física que facilitará el encaminamiento incluso aunque los nodos cambien su localización.

2.2 Intercambio y almacenamiento de archivos

2.2.1 La nube

Durante los últimos años, esta tecnología ha visto su auge y es la opción principal para la mayoría de los usuarios tanto a nivel personal como empresarial mediante servicios como DropBox, iCloud o Google Drive. Relacionando esta tecnología con el punto 2.1.1, esta es la implementación del modelo cliente-servidor para el intercambio y almacenamiento de ficheros. También se utiliza para ofrecer otros recursos de los servidores, pudiendo por ejemplo alquilar parte de su capacidad de computación (Software as a Service).

Este modelo de almacenamiento de datos surgió en los años 1960 [3] y hoy en día es generalmente implementado por grandes compañías con un gran número de servidores. De manera similar a las redes overlay P2P expuestas previamente, el almacenamiento en la nube se lleva a cabo mediante una red lógica superpuesta a la red física, por lo que los archivos pueden estar repartidos por varios servidores en zonas geográficas distantes. Sin embargo todo el entramado será transparente para el usuario, que percibirá lo que para el será un directorio único. Precisamente de ahí surge el nombre esta tecnología, ya que los archivos

estarán repartidos en un número desconocido de servidores que estarán en una situación desconocida.

Figura 11: La Nube

La ventaja principal por la que estos servicios están teniendo tanto éxito es la posibilidad de tener acceso a los archivos desde cualquier dispositivo si se tienen unas credenciales que nos permitan conectarnos a los servidores, ya sea mediante una aplicación o una web; algo que no ofrecen los archivos almacenados de manera local.

Entre otras características este servicio ofrece una tolerancia a fallos muy alta ya que implementa herramientas de redundancia y distribución de copias de los archivos, así como control de versiones de los mismos.

2.2.2 Redes Friend-to-Friend

Actualmente, para intercambio de archivos mediante P2P, lo más utilizado es el protocolo BitTorrent. Hay un gran número de *clientes torrent* que implementan este algoritmo, como por ejemplo µTorrent, BitTornado, etc.

Estos programas hacen uso del protocolo BitTorrent [4] para conectar nodos con archivos que estén dispuestos a compartir. Están tan extendido que existe una gran cantidad de archivos que se comparten entre muchos nodos cuyos usuarios están en puntos opuestos del mundo y nunca tendrán contacto directo. Es decir, los archivos que se intercambian a través de estos programas son, de alguna manera, públicos. Cualquier usuario con uno de estos *clientes torrent* puede descargarse de los demás nodos los archivos que encuentre listados por la web.

Sin embargo, si los usuarios que pueden tener acceso a los archivos que se comparten son un número limitado y determinado, ya no hablamos solo de *Peer-to-Peer* si no de *Friend-to-Friend*. Este término fue acuñado por Dan Bricklin en el año 2000. [5]

Las redes F2F son un tipo concreto de red P2P en la que los usuarios se conectan de manera directa con gente que conocen, asegurando su identidad mediante métodos de autenticación. Así, lo usuarios externos a la red F2F no podrán saber qué archivos se están intercambiando ni qué usuarios son parte de esta red.

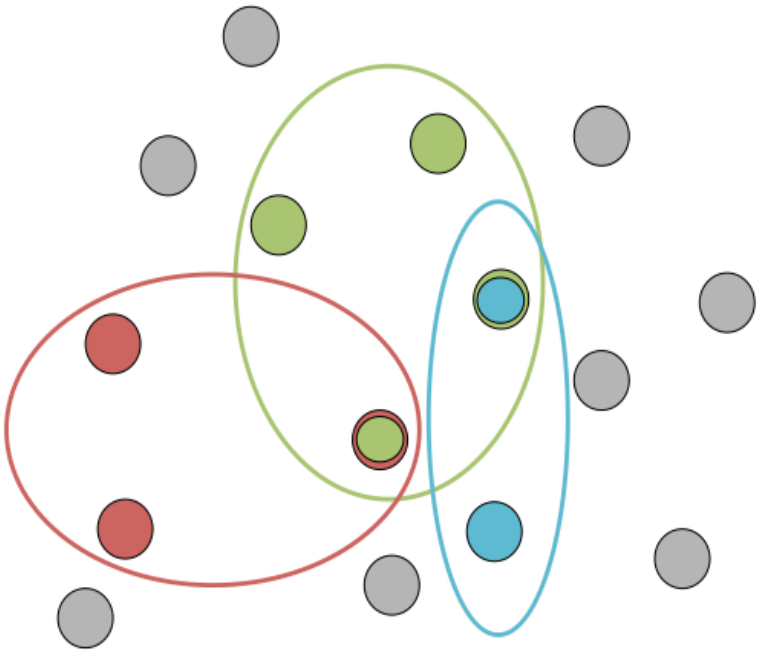


Figura 12: Redes F2F superpuestas

Cada usuario puede pertenecer a varias redes F2F sin que estas interfieran entre ellas, ni tengan acceso a los archivos de las otras redes, son entornos independientes y no relacionados.

Idealmente, el tráfico de la red F2F va cifrado de manera que los archivos que se intercambien no puedan ser comprensibles para nadie que no pertenezca a la red, incluso aunque los interceptara. A diferencia del uso más extendido que se le da a las redes P2P, estas redes se utilizarán para compartir archivos privados, que no serán publicados ni listados en la web.

Existen programas que implementan una red F2F, ya sea como opción complementaria en un cliente P2P normal, o programas que son diseñados exclusivamente para ello. Uno de los más extendidos ha sido Turtle.

Turtle

Aunque ya no se desarrollarán nuevas versiones [6], Turtle ha sido un programa de F2F para Linux que ha disfrutado de gran popularidad. La idea principal detrás de Turtle es que no se den conexiones arbitrarias entre nodos de la red e intercambien archivos, si no que el usuario establece canales seguros y autenticados con un número limitado de nodos controlados por otros usuarios que este conoce.

Tanto las búsquedas como el propio encaminamiento se hace siempre cifrado y salto a salto, resultando en comunicaciones que solamente ocurren entre nodos de confianza. Este enfoque elimina la posibilidad de que un atacante determine qué información se está intercambiando y entre quién.

2.2.3 Sincronización de archivos en redes F2F

Los servicios en la nube pueden darse de manera pública o de manera privada, dependiendo de si se trata de un archivo que cualquier usuario pueda descargarse de un servidor, o si solamente unos usuarios concretos tienen accesos a dicho archivo.

Este segundo caso es al que estamos ya tan acostumbrados en nuestro trabajo diario y será el más similar a las redes F2F, pero además los servicios en la nube de este tipo se han caracterizado por sincronizar automáticamente los archivos disponibles en el directorio privado. Esto es, lo que pongamos en un directorio privado en un servicio en la nube se descargará, o estará disponible, para todos los dispositivos que tengan acceso legítimo a dicho directorio.

Sin embargo estas soluciones en la nube tienen ciertos problemas de seguridad y privacidad. Al subir los archivos a un servidor cuya

localización se desconoce y del que no se tiene control en absoluto, no podremos saber lo que pasa con esos archivos mientras están alojados allí. Esto puede llegar a ser un problema ya que diferentes agentes pueden tener acceso a nuestros archivos sin que nosotros seamos si quiera conscientes.

La respuesta a esta situación en el caso de las redes F2F, será un tipo particular de red en el que se dé esta sincronización pero con una seguridad mucho mayor. De los programas que se han desarrollado en esta línea BitTorrent Sync es quizá el más relevante.

BitTorrent Sync

Este programa desarrollado por los creadores del protocolo BitTorrent está diseñado para sincronizar carpetas compartidas entre usuarios que son parte de una red de F2F, sólo los usuarios que formen parte de la red serán capaces de tener acceso a los archivos.

De cara a la seguridad, este tipo de red será mucho más interesante, ya que, como en Turtle, los archivos viajarán cifrados de nodo a nodo y no se sabrá desde fuera quién está compartiendo qué. La ventaja con respecto a las soluciones en la nube vendrá dada por el hecho de que los archivos estarán alojados localmente en los dispositivos, de los que el usuario tiene el control.

Además, desde un punto de vista más práctico, los programas de sincronización e intercambio de archivos en la nube suelen limitar el espacio disponible para los usuarios. Es así como se monetiza el servicio, ofreciendo más capacidad a cambio del pago de una cuota. En el caso de BitTorrent Sync y los programas de sincronización F2F, la capacidad de la carpeta compartida será tanta como el usuario le conceda dentro de su dispositivo hasta que este se llene.

Capítulo III

3. Diseño

3.1 Requisitos

Como ya se ha explicado de manera breve en el Capítulo I, este proyecto tendrá unos requisitos fundamentales en diferentes áreas. Dichas áreas estarán íntimamente relacionadas, aunque tendrán funcionalidades propias e independientes.

La red estará conformada por nodos que aglutinarán todas estas partes y las permitirán comunicarse e interactuar entre ellas; el nodo se convierte por tanto en la entidad básica principal del esquema del proyecto.

- Deberá poder comunicarse con otros nodos tanto para el intercambio de información, de estructuras de datos, como para el intercambio de archivos.

- También debe ser capaz de mantenerse conectado a otros nodos de manera invisible para el usuario, que no necesitará ejercer tareas de administración de la red.

Para proporcionar el servicio desde cualquier punto de Internet, los dispositivos externos deben poder comunicarse con la red de nodos; esta comunicación se hará a través de la web:

- Dado que las conexiones externas no serán a través de un nodo como los demás, el protocolo de comunicación entre los nodos se desarrollará teniendo en cuenta que deben poder entenderse entre ellos, pero también dar respuesta a las conexiones externas.

Hará falta, por tanto, desarrollar una interfaz web que se conecte e interactúe con los nodos. Deberá ser atractiva para el usuario y que

ofrezca facilidad de uso. Gráficamente no será muy complicado, pero deberá ser funcional en tanto en cuanto todo ocurra de manera transparente para el usuario.

Serán de vital importancia la seguridad y la privacidad. Como eje principal de la motivación de este proyecto se encuentra la de ofrecer el servicio sin que nadie más que los dispositivos de los usuarios involucrados sean los que tengan acceso a la información y los archivos. Por tanto, todos los datos intercambiados deberán moverse en un marco de seguridad y privacidad robusto y completo.

- La información intercambiada, tanto los mensajes que formen parte del protocolo de comunicación como los archivos en sí, deberán ir cifrados para evitar problemas incluso aunque fueran interceptados.

- El eslabón más débil de la cadena será, por supuesto, la interfaz web. Por ello, se implementarán métodos de autenticación que eviten el acceso de personas no autorizadas a los archivos.

- El acceso de un nuevo nodo a la red deberá estar avalado por los nodos que ya formen parte de ella; no se dará acceso a cualquier dispositivo que lo solicite.

3.2 Estructura general

3.2.1 Modelo de red

Los nodos estarán interconectados mediante una red overlay superpuesta a la red IP. Se comunicarán entre ellos e intercambiarán los archivos según el modelo Peer-to-Peer; además, se podrá establecer comunicación con ellos a través de un interfaz web, ya sea para nuestro dispositivo local o para uno remoto. La finalidad principal del servicio será el intercambio y la sincronización de archivos de manera lo más transparente posible para el usuario, sin que sea relevante la localización real de los dispositivos.

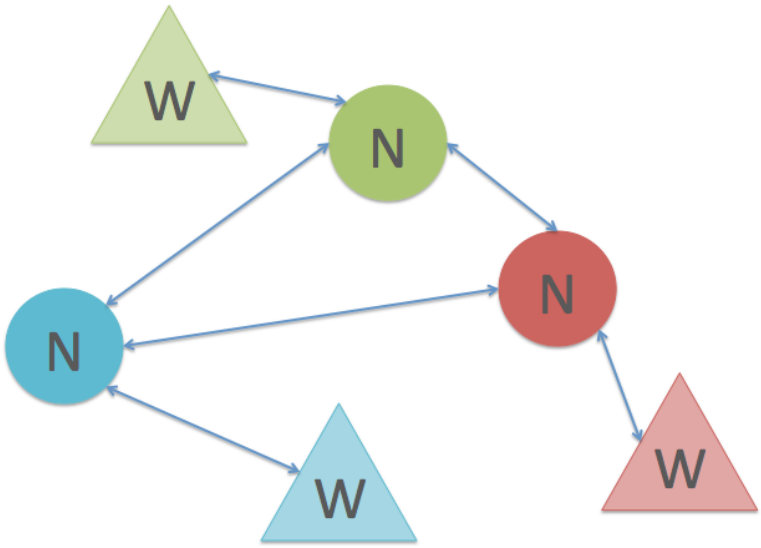


Figura 13: Esquema general de la red.

Se busca que la red sea lo más autónoma posible, incluso en la entrada y salida de nodos. Al ser una red completamente descentralizada y con movilidad de los nodos integrantes la topología será cambiante y la entrada de un nodo nuevo a la red se convierte en un problema que necesita ser resuelto. Surge por tanto la necesidad de, o bien utilizar un tracker ajeno a dicha red, o utilizar un mecanismo de Bootstrapping ofrecido por nodos que ya formen parte de la red.

En el caso del tracker, sería un servicio ofrecido por un servidor externo que listaría las localizaciones y la información necesaria de los nodos presentes en la red. En el caso del Bootstrapping, habrá que hacer una pequeña diferenciación entre nodos Master y nodos normales. No tendrán implementaciones diferentes, pero el nodo que ocupe el papel de Master utilizará unas funcionalidades que en el resto estarán dormidas. El proceso de boostrap consistirá en proporcionar al nodo nuevo la información necesaria acerca de la red overlay para que pueda unirse a ella. En el apartado 3.6 Módulo de Descubrimiento y Bootstrapping se tratará más en detalle esta problemática y se explicará la solución planteada.

3.2.2 Estructura del nodo

En el nodo confluirán todas las partes y las funcionalidades que las comuniquen. Constará de los siguientes módulos y partes, que serán independientes entre ellas:

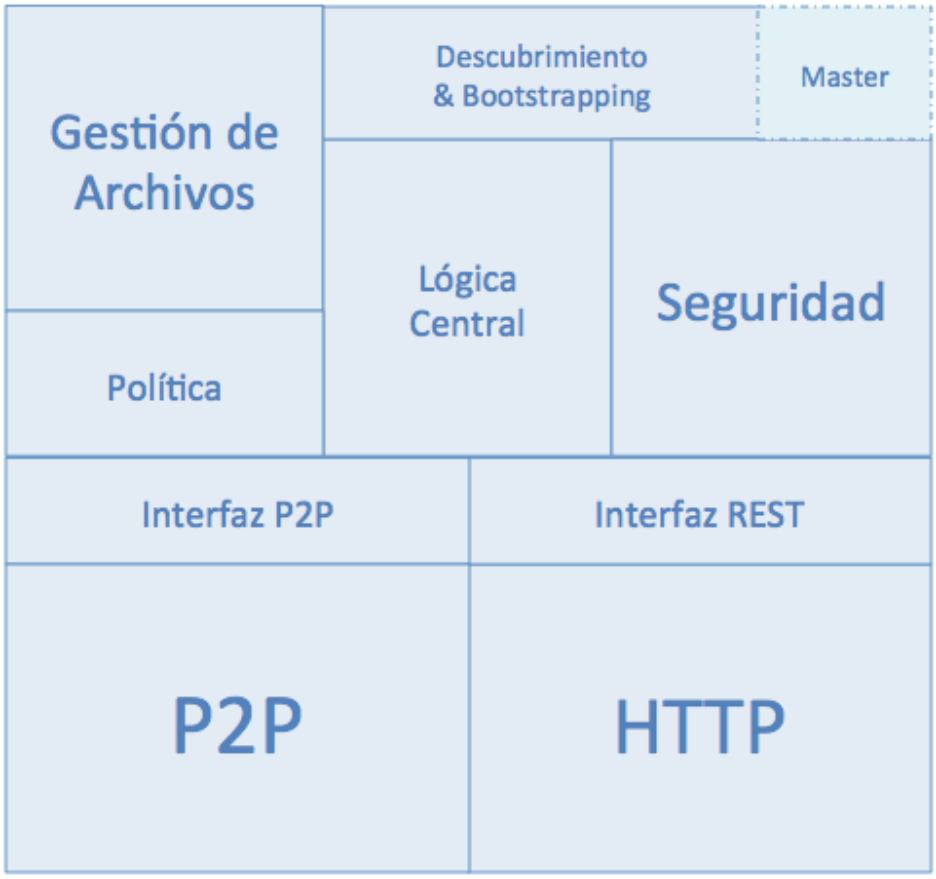


Figura 14: Estructura interna del nodo

3.3 Módulo P2P

Como una de las partes más cercanas a la red IP, tendrá que manejar información al respecto de ésta, siempre garantizando la transparencia para el usuario. En este módulo recaerá también la responsabilidad de conocer la estructura de la red overlay; qué nodos forman parte de la red por el momento, dónde están los recursos y cuales son los identificadores de todos los elementos de dicha red.

Interconexión de los Nodos

Este será el módulo que dote de estructura lógica a la red, conectando los nodos y gestionando el enrutado de paquetes en base a los identificadores de los recursos y los otros nodos de la red. Este enrutado, aunque se realizará evidentemente a través de la red física, no se apoyara en la topología física de la red, si no en la topología lógica de la red overlay. Esto quiere decir que no necesariamente se buscará el camino más corto físicamente, si no el que lo sea en base a los identificadores únicos de los elementos de la red overlay. Un ejemplo posible de esto podemos encontrarlo en el apartado sobre Kademlia [pág. 12]más atrás del Capítulo II sobre el estado del arte.

Se comunicará directamente con los módulos P2P de los otros nodos, y a través del Interfaz P2P con el resto de módulos del nodo. Dicho Interfaz P2P será tal que pueda implementarse el módulo P2P con diferentes clientes P2P sin necesidad de realizar cambios en el interfaz.

Intercambio de Archivos

A través de este módulo se intercambiarán las partes que conformen los archivos. Los archivos no se intercambiarán completos, si no divididos en partes para optimizar los tiempos de descarga y el ancho de banda [pág. 28]. El módulo P2P tendrá información sobre la localización de los otros nodos y funcionalidad para hacerles peticiones de las diferentes partes de archivos, así como para responder a peticiones de descarga que lleguen desde los otros nodos.

Sin embargo la información acerca de qué nodo tiene qué partes no será competencia suya, esa será una de las responsabilidades del módulo de Gestión de Archivos. Asimismo, las peticiones de descarga llegarán y se enviarán a través del modulo HTTP, el módulo P2P solamente se ocupará de darles respuesta.

3.4 Módulo HTTP

Este módulo también estará colocado justo encima del nivel de red. A través de él se moverá el flujo de peticiones entre los nodos, así como el intercambio de información necesaria sobre los archivos. Se comunicará tanto con el módulo HTTP de otros nodos como con las interfaces web que a las que accederá el usuario a través de un navegador de Internet.

Comunicación Entre Nodos e Interfaz Web

El módulo HTTP gestionará las peticiones de descarga de unos nodos a otros, así como información acerca de los listados de archivos que cada nodo tiene en su poder, aunque no procesará dicha información. Controlará por tanto el flujo de información necesario para el correcto funcionamiento en el intercambio de archivos a nivel interno en la aplicación.

De la misma manera, podrá recibir peticiones de los interfaces web desde fuera de la red de nodos. Dicha interfaz web podrá ser la correspondiente al mismo nodo en el que se encuentre el módulo HTTP o el de otro nodo que pertenezca a la red. Esto esta pensado para la transparencia de uso y localización, de manera que el usuario pueda acceder a los archivos de la misma manera si lo hace desde su propio dispositivo o desde un navegador remoto.

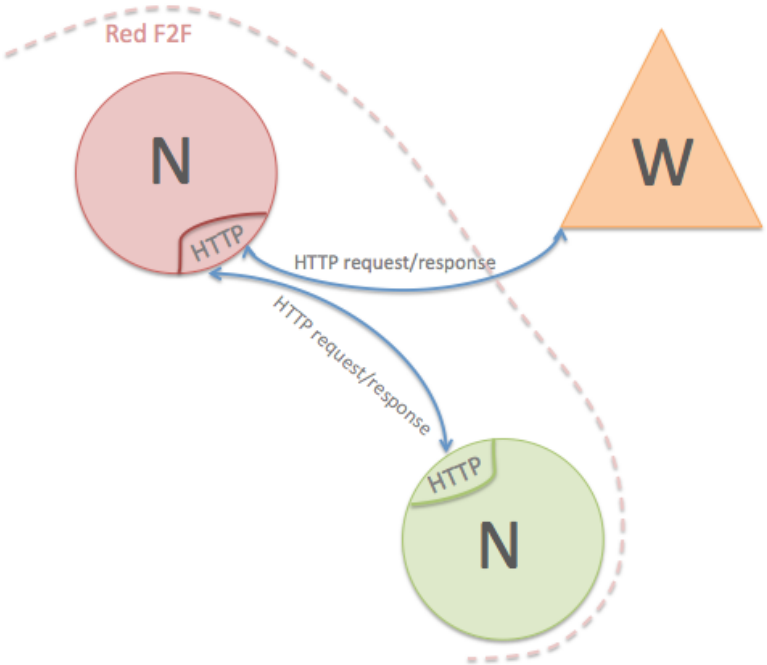


Figura 15: Comunicación HTTP entre Nodos/Interfaz Web

Jugará el mismo papel que el Interfaz P2P en el módulo P2P, pero para las peticiones HTTP. La ventaja de este tipo de interfaz es la posibilidad de comunicar los navegadores web con los nodos normales mediante el mismo tipo de funciones. Se comunicará con el Interfaz P2P, con el módulo de Gestión de Archivos y con la lógica principal de la aplicación.

3.5 Módulo de Gestión de Archivos

Este será el módulo que se ocupará de todas las operaciones y funcionalidades relacionadas con los archivos y los listados de los mismos. Jugará un papel activo de comunicación entre el módulo HTTP, del que se recibirán las peticiones de información sobre el listado, el módulo P2P, que se ocupará de realizar esas descargas y envíos, y se ocupará de la gestión de los archivos y sus partes a nivel local en el dispositivo.

Gestión de Partes de los Archivos

Los archivos no serán compartidos enteros; siguiendo el ejemplo del protocolo BitTorrent, se dividirán los archivos en partes de tamaño fijo. Una vez el nodo tenga todas las partes de un determinado archivo, el módulo de Gestión de Archivos será quien se ocupe de re ensamblar todas las partes en el archivo completo.

Mediante este mecanismo, será fácil saber qué porcentaje de la descarga se ha completado, y se optimizará el ancho de banda y los tiempos de descarga de los archivos. Dado que las velocidades de subida son en la mayoría de los casos más bajas que las de descarga, descargar diferentes partes de varios nodos a la vez permitirá que todo el ancho de banda de bajada se ocupe, mientras que no se formarán cuellos de botella de salida de los nodos emisores.

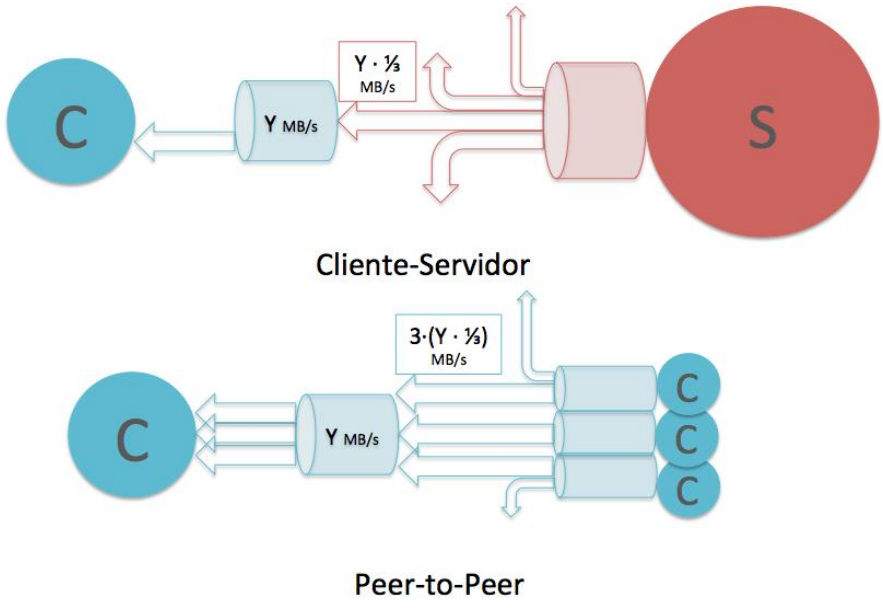


Figura 16: Ancho de banda Cliente-Servidor VS P2P

Este será también el módulo que se ocupe de llevar un seguimiento de los archivos que el usuario tiene en el dispositivo, así como de las partes se han descargado ya.

Con esta finalidad, se creará una estructura de datos específica en la que estará toda la meta información necesaria. Después de reflexionar acerca de cuál sería exactamente la información más relevante se ha pensado en los siguientes campos sobre cada archivo del directorio local:

- Nombre del archivo:
  Deberá ser único ya que en base a él se calculará un identificador en el módulo P2P para su descarga.

- Formato del archivo:
  Será útil en caso de realizar una búsqueda de un tipo de archivo, o incluso si se quisiera filtrar solamente un determinado tipo de archivos.

- Número total de partes del archivo:
  Al dividirse siempre los archivos en partes de igual tamaño, será inmediato conocer el número total de partes en las que se divide un archivo. Servirá para conocer cuantas partes de un archivo nos faltan por descargar.

- Sublistado de partes descargadas:
  Se listaran, por nombre, las partes ya disponibles localmente de un determinado archivo. Se decidirá una nomenclatura para relacionar un archivo con sus partes.

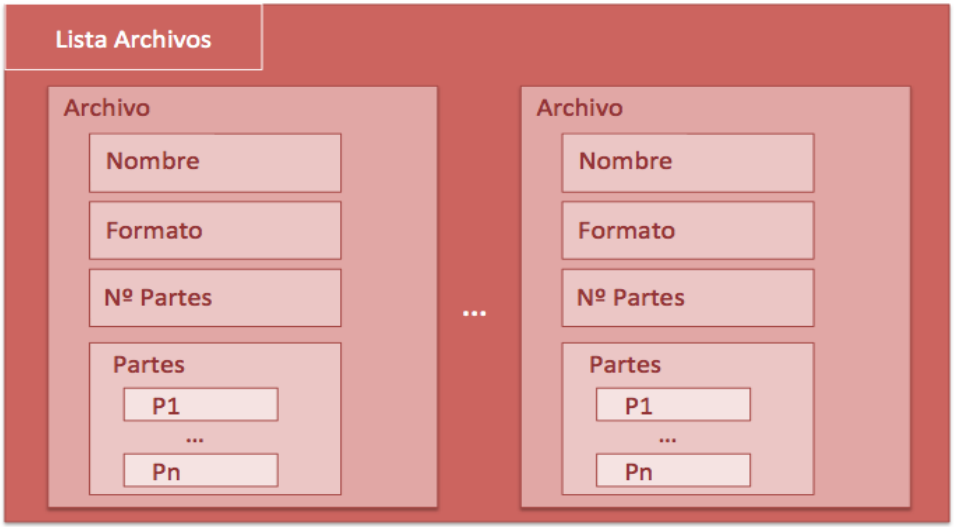


Figura 17: Estructura de datos del listado

El listado deberá estar actualizado constantemente, por lo que siempre que haya un cambio deberá editarse. Esta comprobación se dará en uno de los siguientes escenarios:

Al iniciar la aplicación

Si el usuario ha borrado algún archivo del directorio de descargas mientras la aplicación estaba apagada, se eliminará esa referencia del listado. Análogamente, si al iniciar la aplicación hay archivos nuevos que el usuario quiera compartir, se sacará toda la información importante del archivo y se listará.

Al descargar un archivo

Cuando el usuario decide descargar un archivo, se añadirá una referencia a dicho archivo en el listado y se irá rellenando la referencia a medida que las diferentes partes sean descargadas.

Al eliminar un archivo

Si se decide eliminar algún archivo del directorio mientras la aplicación está en marcha, su referencia en el listado debe ser eliminada también.

Al cerrar la aplicación

Este caso no es estrictamente necesario ya que si en todos los otros escenarios previos se ha actualizado correctamente el listado, éste debería estar completo en el momento del cierre de la aplicación. Sin embargo, en aras de que el sistema de listado sea más robusto, se hará una última comprobación del directorio antes del cierre.

Control del Intercambio de Archivos e Información Sobre Archivos Disponibles

En lo respectivo a las descargas, este módulo estará siempre en comunicación con el módulo P2P. Le dará instrucciones cuando se añada un archivo o partes de él para que las ponga a compartir y que los otros nodos puedan descargarlas. También, en el caso contrario en que un archivo sea eliminado por el usuario a nivel local, informará al módulo P2P para que elimine ese archivo de la circulación.

El módulo de Gestión de Archivos tendrá comunicación regular con el módulo HTTP, de quien recibirá consultas acerca del estado de su listado de archivos. Cada nodo estará al tanto de lo que tienen el resto de los nodos mediante estas peticiones gestionadas por el API REST; con esa información decidirán qué partes pedir a qué nodos para optimizar sus descargas.

Como hemos dicho antes, se creará una estructura de datos específica para el listado de datos. Normalmente las peticiones de información desde otros nodos no estarán referidas a todos los archivos que tenga disponible nuestro usuario, por lo que no tendrá sentido enviar la estructura de datos completa.

Antes de enviar la información, el módulo de Gestión de Archivos se ocupará de extraer únicamente la información solicitada y le pasará al módulo HTTP una versión mucho más simple. Si, por ejemplo, el nodo al que le solicitásemos la información tuviera tres archivos disponibles, pero a nosotros solamente nos interesara el primero de ellos, se extraería solamente la información de las partes del primer archivo.

3.6 Módulo de Descubrimiento y Bootstrapping

Como se explica brevemente en el apartado 3.2.1 Modelo de red, dada la naturaleza distribuida y descentralizada de la red del proyecto, hará falta algún tipo de sistema para que los nodos sean capaces de conocer la red y entrar en ella. La opción del tracker queda desechada en aras de que la red no dependa en absoluto de partes ajenas a la propia red.

Bootstrapping con el Nodo Master

El módulo de Bootstrapping, incluido en todos los nodos aunque no este activo, tendrá dos partes: la que utilizarán los nodos que quieran entrar en la red y la que usará el nodo Master para dar respuesta a esos nodos que quieran entrar.

El nodo Master enviará la información necesaria acerca de la topología y la composición de la red al nodo nuevo. Mediante dicha información, el nuevo nodo que esté intentando entrar a la red conocerá la topología de esta y podrá comunicarse con el resto de nodos.

Este proceso no será complicado ni largo pero deberá ser seguro, ya que será el punto de entrada de los nodos en la red. Los mecanismos de identificación explicados en el siguiente apartado (3.7 Módulo de Seguridad)jugarán su papel durante el proceso de Bootstrapping.

Descubrimiento de la Red

Antes de entrar en contacto con el nodo Master para llevar a cabo el Bootstrapping, los nodos nuevos tendrán que ser capaces de comunicarse con él. Para conseguirlo, la dirección IP de este nodo será o bien fija, o bien se utilizará un servicio de DNS Dinámico, de manera que se actualice automáticamente la información de dominio.

La IP fija será útil solamente si la red es muy pequeña y uno de los nodos (el Master) se queda siempre en el mismo sitio. Para redes más grandes, o con movilidad de todos los nodos, será más conveniente utilizar el servicio de DNS dinámico. Los nodos se referirán a un nombre y el servicio se ocupará de devolverles el dominio específico en el que se encuentra ese nodo Master en ese instante.

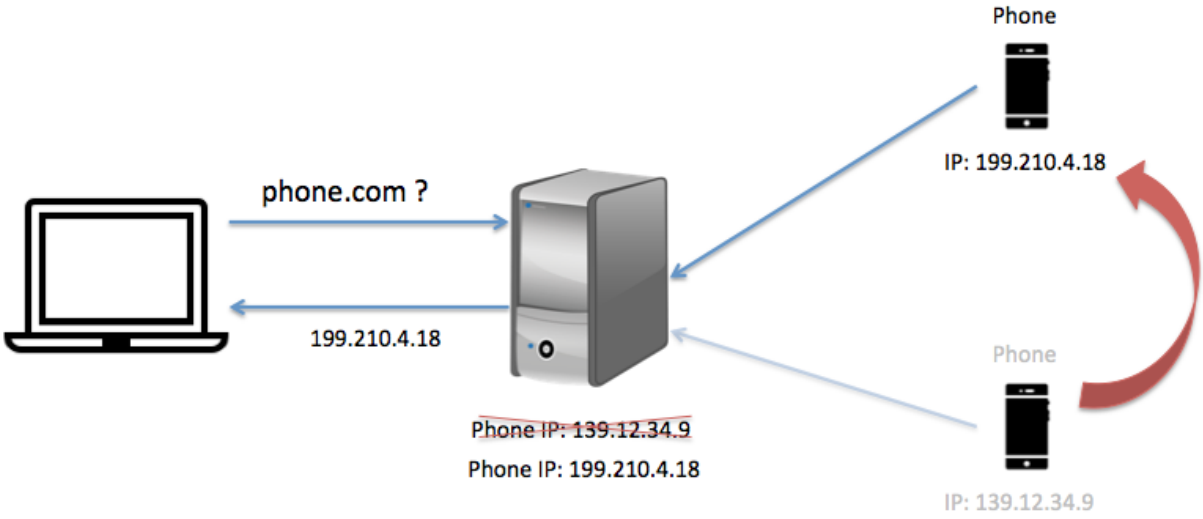


Figura 18: DNS dinámico

Además, para que no haya dualidades, en cualquier momento habrá únicamente un nodo Master presente en la red. Si el nodo Master actual tuviera que abandonar la red, cualquier otro nodo puede heredar sus responsabilidades y convertirse en el nuevo nodo Master, de manera que la red siempre sea accesible desde fuera.

3.7 Módulo de Seguridad

Aunque serán necesarios muchos otros aspectos para tener una aplicación funcional, la motivación principal de este proyecto se centra en desarrollar un servicio de intercambio y sincronización de archivos que sea más seguro y privado que los actuales servicios basados en el modelo Cliente-Servidor.

Este módulo ocupará un lugar cercano a otros módulos que tengan contacto directo con el resto de la red, ya que será por donde pueda estar amenazada la aplicación. Es decir, la actividad del módulo de seguridad afectará principalmente al módulo P2P a través del cual se enviarán los archivos, al módulo HTTP, que intercambiará información acerca de los archivos disponibles, y al módulo de Descubrimiento y Bootstrapping, que podrá incorporar nodos nuevos a la red.

Identificación de Nodos a la Entrada a la Red

Al tratarse de una red Friend to Friend, como caso específico de red P2P, solamente podrán entrar en la red nodos que sean aceptados por los integrantes de la red ya formada. Para simplificar las cosas y facilitar la identificación, serán los nodos que integren la red los que envíen invitaciones de entrada a los nodos nuevos. Además, asumiendo que los nodos confían los unos en los otros, con que uno de ellos le envíe una invitación a un potencial nodo entrante será suficiente. El proceso de Bootstrapping no ocurrirá hasta que el nodo entrante haya sido identificado correctamente.

La invitación deberá contener información acerca del servicio DNS dinámico para que el nodo entrante pueda ponerse en contacto con el nodo Master con quien iniciará el proceso de Bootstrapping. Esta invitación se compartirá con el nodo entrante por medios ajenos a la propia aplicación; de esta manera se enviará específicamente al nodo entrante y solamente él. Esta invitación tendrá fecha de caducidad, después de un tiempo especificado ya no tendrá validez ni si quiera para el nodo para el que se emitió.

Como medida adicional, una vez el Master haya iniciado el proceso de Bootstrapping con el nodo entrante, consultará a los nodos que ya formen la red acerca de cual de ellos le invitó a formar parte del grupo. En caso de que ninguno de los nodos responda positivamente, el Master finalizará el Bootstrapping y el nodo entrante quedará fuera de la red.

Para conseguir que el intercambio y la gestión de archivos en el programa sea lo más parecida posible a una gestión local, solamente el usuario puede tener acceso a sus archivos. Por lo tanto, toda la información que circule entre los nodos deberá cifrarse para que los individuos ajenos al programa no la comprendan.

La criptografía asimétrica, basada en el cifrado y descifrado de los mensajes mediante parejas de claves pública y privada parece la opción más conveniente para este proyecto. En el momento de añadir un nuevo nodo a la red, en el momento del Bootstrapping, se intercambiarán también las claves públicas de los diferentes nodos. Si siguiéramos una analogía de llaves y candados, la clave privada sería la llave y la pública el candado.

Cuando los nodos quieran mandar información a otro nodo, ya sea una de las partes de algún archivo o el listado de archivos que tiene disponibles, la cifrará utilizando la clave pública del destinatario. Cuando la información llegue al nodo de destino, este podrá descifrar los paquetes y actuar como proceda con la información.

Siguiendo con la analogía anterior, cuando se cifra con la clave pública, la información habría sido guardada en una caja a la que se le hubiera puesto un candado. Una vez esto haya ocurrido, ni si quiera el emisor podría volver a abrir la caja, ya que únicamente el nodo de destino conoce la clave privada, es decir, es el único que tiene la llave que abre ese candado.

Este sistema permite que, incluso ante un posible ataque en el que algún individuo intermedio intercepte los paquetes sin permiso, no se pudiera descifrar la información y siguiera siendo privada.

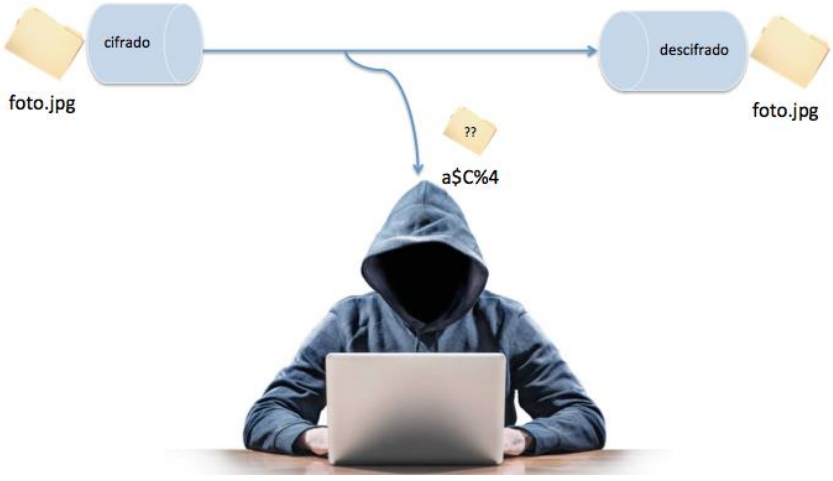


Figura 19: Cifrado entre nodos

Este enfoque solucionaría el problema en caso de que el tráfico no fuera demasiado grande si, por ejemplo, hubiera pocos nodos en la red; en caso de que la red creciera mucho se podría optar otra alternativa, ya que el coste computacional de cifrar mediante la clave pública puede llegar a ser alto. En este nuevo escenario nada cambiaría a la hora del Bootstrapping y el intercambio de claves públicas iniciales. Sin embargo, el cifrado público no se utilizaría para todos los mensajes, si no solamente para poder intercambiar las claves de sesión que se utilizarán a partir de ese momento.

La criptografía asimétrica funcionaría igual que la asimétrica ya explicada, pero se cifraría y descifraría con la misma clave. Por eso no se podría intercambiar la clave en claro y haría falta el uso de la criptografía asimétrica para que todos los nodos la consiguieran de manera segura. Una vez todos los nodos tengan la clave simétrica de sesión, podrán intercambiar los mensajes cifrados sin peligro de que los mensajes sean legibles incluso aunque fueran interceptados.

Autenticación de los Nodos Pertenecientes a la Red

Al superponer la red overlay lógica a la física, los nodos estarán separados en la red IP de manera aleatoria, es decir, los mensajes viajaran por Internet de un nodo a otro, no es una red cerrada. Esto implica que puede haber nodos no pertenecientes a la red que estén al tanto de que hay unas ciertas comunicaciones entre ellos, e intenten suplantar sus identidades con fines maliciosos.

Para combatir este problema, los nodos de la red utilizarán mecanismos de autenticación que los identificarán entre ellos y se darán cuenta de si los mensajes que reciben no vienen de componentes del grupo. De nuevo, la criptografía asimétrica será la solución a este problema de seguridad.

Los mensajes llevarán un identificador único que será calculable por el receptor: el identificador será un resumen del propio mensaje, calculado mediante técnicas de hash. Este identificador del mensaje se cifrará antes del envío mediante la clave privada, única del emisor y solo conocida por él. A la llegada al receptor, éste descifrará el identificador cifrado y calculará él mismo mediante la misma técnica de hash el resumen del mensaje. Así, podrá comparar el calculo obtenido con el identificador del mensaje. En caso de que coincidan podremos estar seguros de que el emisor es quien dice ser, ya que solamente él tiene en su poder la clave privada que cifró ese identificador.

3.8 Módulo de Política

Este módulo, aunque bastante pequeño, jugará un papel importante a la hora del intercambio de archivos, donde afectará principalmente. Como módulo de control entre el módulo de Gestión de Archivos y el módulo P2P, esta pequeña parcela se encargará de aplicar políticas predefinidas al intercambio de archivos, pudiendo alterar así el número de archivos descargados de un nodo concreto o parando la descarga de dicho nodo hasta que se cumplan determinadas condiciones.

Para aclarar cuales pudieran ser estas políticas de intercambio de archivos se consideran algunos posibles escenarios. Un hipotético caso sería aquel en el que uno de los nodos sea un teléfono móvil sin acceso a Wi-Fi y con una tarifa reducida de datos; la descarga podría, por ejemplo, pausarse hasta que el teléfono estuviera conectado a una red Wi-Fi.

Otro posible escenario sería uno en el que uno de los dispositivos que contenga un nodo de la red este bajo de batería y sin embargo tenga archivos que nos interese descargar. En este caso, podrían implementarse políticas de balanceo de descarga que disminuyeran, o incluso anularan, el número de partes de dichos archivos que se le solicitan a ese nodo, pasando a ser descargados de los nodos restantes.

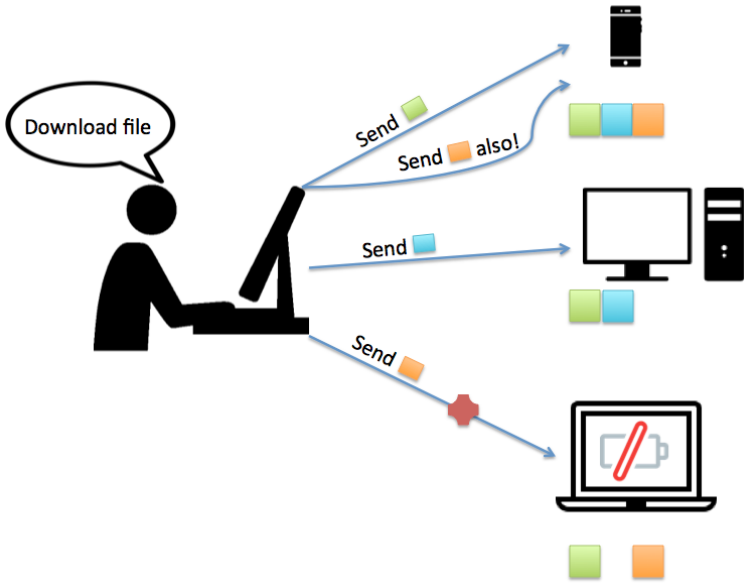


Figura 20: Versatilidad en la descarga de archivos según políticas aplicadas

En cualquier caso, qué políticas específicas será más conveniente aplicar se decidirá en el momento del desarrollo del proyecto según las necesidades encontradas.

3.9 Módulo Central de la Aplicación

El módulo central será importante, ya que estará por encima de todos los demás módulos controlando el work-flow normal de la aplicación. No tendrá un número muy alto de funcionalidades, pero las que tenga serán muy necesarias para el correcto funcionamiento de todo el programa.

De lo primero que se ocupará será de iniciar todos los demás módulos en el momento de encendido de la aplicación. Dará orden de actualizar la lista de archivos disponibles, avisará al módulo P2P de que comparta lo que deba, etc. Durante la actividad de la aplicación tendrá menos actividad, una vez ya puedan empezar a comunicarse los módulos entre ellos, solamente se ocupará de mantener la aplicación viva. Volverá a jugar un papel clave en el momento del cierre de la aplicación, dejando todo bien atado y llevando a cabo los procesos necesarios para que cuando se vuelva a abrir todo esté correcto.

3.10 Interfaz Web

Aunque la Interfaz Web no es un módulo como tal ya que no forma parte de la estructura del nodo, será la manera en la que el usuario interactuará con otros dispositivos de la red, así como con su propio nodo. La idea detrás de esta configuración es que, siguiendo la línea de transparencia de uso para el usuario, sea indiferente para él acceder a archivos de su propio dispositivo o de un dispositivo remoto, en ambos casos solo necesitará utilizar la Interfaz Web.

Cada Interfaz Web estará asociada a un nodo concreto, que será quien se comunique con los otros nodos y lleve a cabo la gestión de archivos. Se seguirá este diseño dado que los navegadores web no son tan potentes como lo serán los propios nodos; así el Interfaz Web tendrá mucha menor carga de trabajo y su desempeño será más fluido.

Por ejemplo, si el usuario se conectase desde un dispositivo que no perteneciera a la red (i.e. quisiera descargar un archivo desde un ordenador de la Universidad) podría hacerlo y tener acceso a los archivos igual que ocurre en los actuales servicios de alojamiento de archivos en la nube. Esta descarga ocurriría a través de su nodo asociado, quien descargaría todas las partes del archivo a través del módulo P2P, las re ensamblaría en el archivo íntegro y enviaría este a través de HTTP al dispositivo desde el que se conecta el usuario.

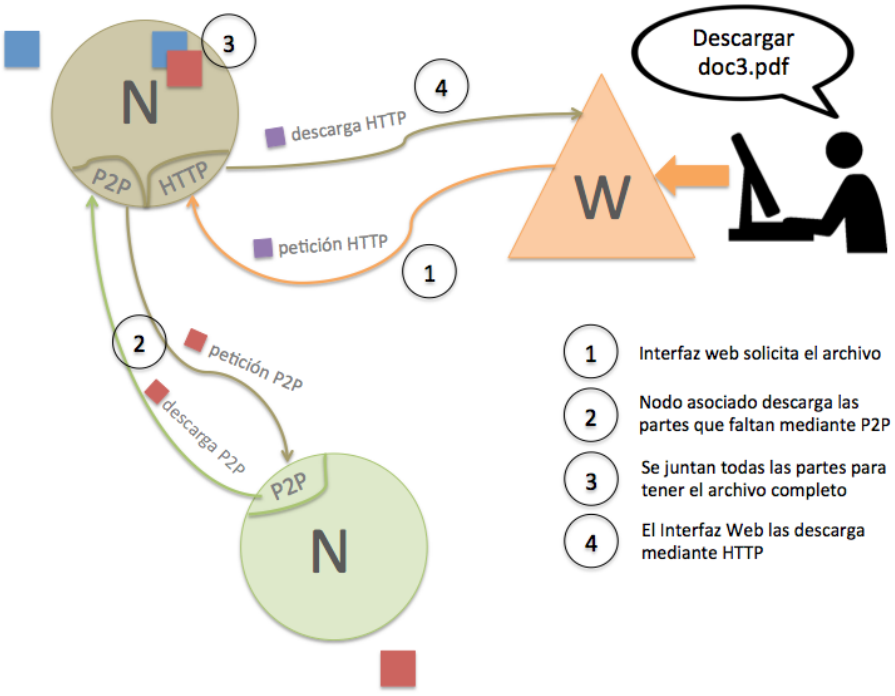


Figura 21: Descarga de archivo a través del Interfaz Web

El Interfaz Web constará de una capa gráfica, inicialmente no demasiado complicada, en la que el usuario encontrará un menú con las diferentes opciones que pueda querer llevar a cabo. Entre ellas se contarán: la obtención de información acerca de qué archivos hay disponibles; la posibilidad de descargar uno, varios, o todos los archivos; el borrado de un archivo seleccionado; etc.

Por supuesto, para evitar que personas no deseadas tuvieran acceso a los archivos mediante la conexión a los nodos a través del Interfaz Web, se implementaría un mecanismo de autenticación mediante credenciales (nombre de usuario y contraseña) análogamente al caso de los servicios en la nube tradicionales.

Capítulo IV

4. Desarrollo

4.1 Desarrollo General del Proyecto

La implementación de este proyecto se ha llevado a cabo con el desarrollo de código propio, así como con la utilización de código ya existente con licencia de código abierto. Se justifica en este capítulo la elección entre las diferentes posibilidades, así como se referencian las partes del proyecto que han sido desarrolladas por terceros.

Se ha desarrollado en Java toda la estructura del nodo; para lo concerniente a la Interfaz Web se ha utilizado JavaScript y HTML. Con respecto a Java, se ha elegido este lenguaje de programación dados los conocimientos previos del autor y por el extenso soporte y código disponible en Internet del que se podía hacer uso. En lo referente a JavaScript y HTML, eran las opciones más convenientes para desarrollar una página web con funcionalidades más allá de lo puramente estético.

Se ha conseguido llegar a una versión en la que las funcionalidades implementadas permiten un uso suficiente del programa, en el que los objetivos de la idea principal se han visto satisfechos. Sin embargo, debido al tiempo limitado y la complejidad de algunos de los puntos, quedarían aún por implementar algunas funcionalidades. En el futuro podrían desarrollarse para completar el planteamiento completo de la idea de un sistema de intercambio de archivos privado sin necesidad de servidores ajenos a los usuarios.

4.2 Módulo P2P

Como se explica en el Capítulo de Diseño, el módulo P2P puede funcionar con cualquier implementación de Peers P2P, siempre que pueda cumplir las funciones provistas en el Interfaz P2P. En un primer momento se optó por utilizar Ttorrent [7], dada la simplicidad de uso de sus librerías y el fácil acceso a ejemplos y documentación. Desafortunadamente, cuando se llegó a un primer código funcional surgió el problema del tracker. Ttorrent basa sus funcionalidades de descubrimiento de la red en un tracker; esta solución se descartó, por lo que con ella quedó descartada la posibilidad de usar Ttorrent.

En su lugar se buscó una implementación, también en Java, que no hiciera uso de tracker y ofreciera la posibilidad de efectuar el descubrimiento de la red mediante Bootstrapping. Estudiando diferentes implementaciones que utilizaban los protocolos ya existentes, se encontró TomP2P: una implementación de una DHT basada en Kademlia [8] que utiliza métodos de Bootstrapping y tenía instrucciones y ejemplos extensos y útiles. Aunque no tan intuitiva como Ttorrent, TomP2P es una implementación potente que daba respuesta a las necesidades de este prototipo. Otra ventaja por la que se decidió utilizar TomP2P fue su compatibilidad con Android, así como ejemplos de utilización y configuración, lo que será útil en versiones futuras del proyecto.

Interconexión de los Nodos

En TomP2P los Peers son la entidad básica, igual que los nodos son la entidad básica de este proyecto. A través del módulo P2P se mantiene la red unida, es decir, son los Peers dentro de cada nodo los que mantienen la comunicación abierta con los otros Peers de la red. Si un Peer se desconecta, los nodos adyacentes se darán cuenta e interrumpirán las comunicaciones con él.

A través de estos Peers, TomP2P se ocupa automáticamente de mantener la red overlay viva y del enrutado de los paquetes a través de esta, por encima de la red IP. Los Peers tendrán una tabla de enrutado con los Peers vecinos, donde la cercanía esté referida a la red lógica y no a la física. El enrutado se hará de Peer a Peer, por lo que no se necesitará conocer la localización de todos los demás para hacer llegar paquetes al otro lado de la red (DHT)

Los archivos, o más concretamente sus partes, se intercambian a través de TomP2P. Si un nodo es informado por su módulo de Gestión de Archivos de que tiene disponible una parte de un archivo, el módulo P2P lo pondrá a compartir. Esto no significa que automáticamente se lo envíe a todos los demás Peers que conforman la red, si no que les anuncia que tiene disponible esa parte para descarga. Si más tarde alguno de los otros Peers recibe la orden de descargar esa parte, sabrá de quién puede hacerlo.

Estos métodos de anuncio y descarga de datos se implementan en TomP2P mediante las funciones '*put'* y *'get'* respectivamente. Salvo que se especifique explícitamente de quién se debe efectuar la descarga, la lógica propia de TomP2P elegirá a quién pedírselo en base a los identificadores y las distancias de la red overlay.

Las acciones de anuncio ('*put'),* ocurren siempre que un archivo este disponible en un nodo. Al iniciar la aplicación, entre las actividades iniciales, el módulo P2P anuncia todos los archivos que tiene disponibles después de que el módulo de Gestión de Archivos lo compruebe. De igual manera, si un archivo es eliminado del directorio por el usuario, el módulo de Gestión de Archivos informará al módulo P2P y ese archivo se retirará de los anunciados por ese Peer. Además, cada vez que se descarga algo satisfactoriamente ese Peer pasa a compartir también esa parte de un archivo, aumentando así el abanico de posibles descargas de esa parte.

4.3 Módulo HTTP

Para el módulo HTTP, se ha decidido utilizar un framework llamado Jersey. Este framework de código abierto desarrollado por Oracle tiene una extensa documentación, es fácil de utilizar y ofrece una forma potente de desarrollar servicios web mediante APIs REST. Además, Jersey ofrece dos características que han sido útiles a lo largo del desarrollo del proyecto.

La primera es la posibilidad de implementar, además del servidor, el cliente REST. Esto fue útil en un principio, antes de tener desarrollada el resto de la estructura de los nodos, para probar el framework y ver como interactuarían estos con el módulo HTTP.

La segunda característica interesante de Jersey es el soporte de una gran variedad de estructuras en los mensajes. Jersey efectuará técnicas de marshalling sobre estas estructuras, transformándolas para que sean fácilmente manejables en Java. En concreto, en este proyecto se utilizan mensajes de tipo JSON; esta estructura de datos es la más utilizada en comunicaciones web entre servidores y navegadores. Por esta razón y por lo intuitivo de su uso, se ha elegido JSON como estructura de datos para la comunicación entre el Interfaz Web y el módulo HTTP de los nodos.

API REST

El API REST, como ya se explicó en el capítulo 3.4 Módulo HTTP, será la pieza clave que permita una comunicación igual entre Interfaz Web–Nodo y entre Nodo–Nodo. El Nodo jugará el papel de servidor web para con el Interfaz, por tanto tendrán una serie de funciones para dar respuesta a las diferentes acciones que el usuario pueda llevar a cabo desde dicha Interfaz Web.

Como función más básica, se encargará de servir los archivos necesarios para que el navegador le presente al usuario una página web atractiva y funcional. Estos archivos incluirán los HTML, JavaScript y CSS necesarios.

Además, para que el usuario sea capaz de interactuar con la aplicación y descargar y gestionar los archivos que quiera, necesita saber qué archivos tiene ya disponibles en su dispositivo y qué archivos tienen el resto de nodos. Estas dos funciones se llevarán a cabo sin necesidad de que se conozca la localización exacta de esos dispositivos. Una vez el usuario

disponga de esta información podrá descargar mediante HTTP el archivo que elija.

En una primera versión el nodo y el navegador intercambiaban esta información en forma de archivo XML. Sin embargo, a medida que el proyecto avanzó se pudo ver que esta solución, aunque funcional, era altamente ineficiente ya que compartía grandes porciones de información sobre otros archivos que no había sido solicitada. Como mejora en posterior, y como se ha menciona en las características del servidor Jersey, se selecciona en el nodo la información específica solicitada y se envía al navegador en formato JSON, que luego será presentada de una manera más atractiva mediante el Interfaz Web.

La función de descarga será la que se ocupe de enviar por HTTP el archivo seleccionado al navegador web del dispositivo desde el que se conecte el usuario. Esta función tendrá dos actuaciones posibles: en caso de que el modulo de gestión de archivos le informe de que el archivo se encuentra disponible localmente, la enviará directamente al navegador; en caso de que el archivo se encuentre en otro dispositivo, primero se descargará ese archivo en el nodo a través del módulo P2P para ser enviado por HTTP al navegador una vez sea re-ensamblado.

4.4 Módulo de Gestión de Archivos

Gestión de Partes de los Archivos

Para optimizar la descarga de los archivos, se dividirán en partes y se compartirán las partes por separado. Por tanto, hacía falta una función que dividiese los archivos en trozos de tamaño fijo y que los nombrase según una nomenclatura específica.

En la función *'splitFile'* desarrollada, se irá leyendo el archivo en tramos de 1MB que se separarán en archivos sin extensión. Se habrán convertido básicamente en ristras de bytes de 1MB de longitud. Para poder gestionar dichas partes en operaciones futuras, se decide que la nomenclatura para nombrarlas sea el nombre original del archivo que se divide (cuyo nombre debe ser único en la aplicación) seguido de "_partX" donde X representa el número de parte.

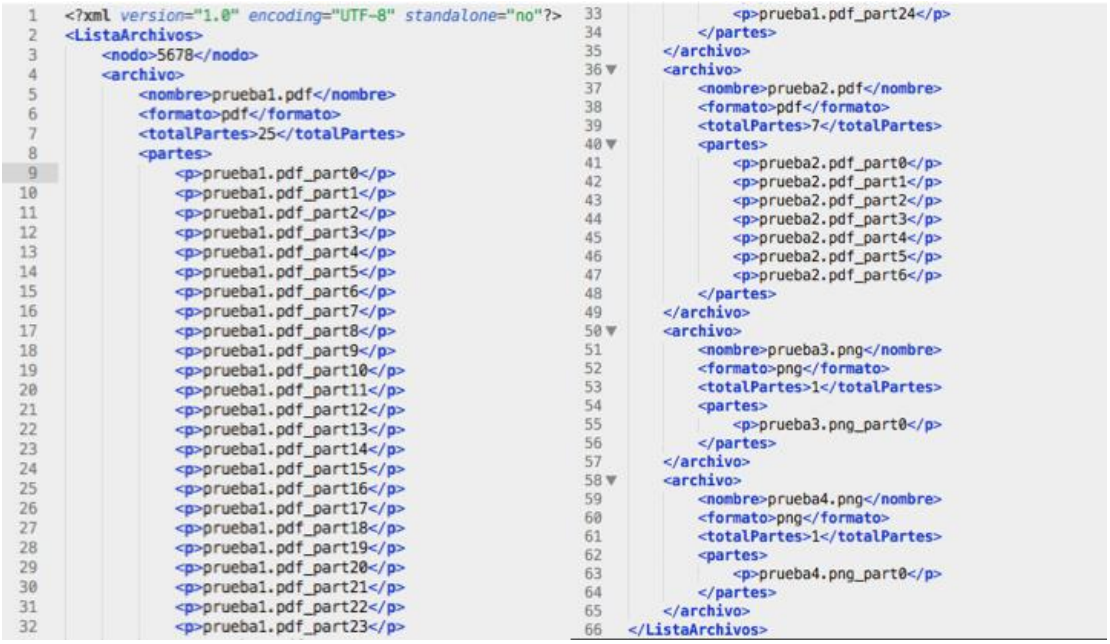
Por ejemplo, un archivo llamado "song.mp3" de 2,8 MB se dividirá en dos partes de 1MB ("song.mp3_part1", "song.mp3_part2") y en una parte de 0,8 MB ("song.mp3_part3"). Si otro de los nodos descargara todas las partes, se volverían a juntar conformando de nuevo el archivo "song.mp3" mediante la función *'rejoinFile'*.

Listado de Archivos Disponibles
XML Local & Objeto Lista

Será esencial para el correcto funcionamiento del módulo de Gestión de Archivos saber qué archivos se encuentran disponibles a nivel local en las carpetas de la aplicación. Para esto se ha desarrollado la función '*whatFilesIn'* con la que se evaluara tanto qué archivos completos hay en la carpeta, como qué partes de los diferentes archivos se tienen. Con esta información, que se actualizará periódicamente durante la ejecución de la aplicación, se actualizará el listado de archivos y partes disponibles.

Este listado tendrá dos representaciones, aunque conceptualmente serán idénticos y tendrán la misma estructura. La primera estructura de datos con la que se trabaja es un archivo XML. Este archivo formará parte de los necesarios para el correcto funcionamiento del programa, especialmente al inicio del mismo, ya que será el archivo que documente el estado de los archivos en el momento del último cierre de la aplicación. Esto se conseguirá mediante la utilización de XPath [9], un lenguaje para

construir y recorrer archivos XML mediante expresiones sencillas e intuitivas.



```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<ListaArchivos>
    <nodo>5678</nodo>
    <archivo>
        <nombre>prueba1.pdf</nombre>
        <formato>pdf</formato>
        <totalPartes>25</totalPartes>
        <partes>
            <p>prueba1.pdf_part0</p>
            <p>prueba1.pdf_part1</p>
            <p>prueba1.pdf_part2</p>
            <p>prueba1.pdf_part3</p>
            <p>prueba1.pdf_part4</p>
            <p>prueba1.pdf_part5</p>
            <p>prueba1.pdf_part6</p>
            <p>prueba1.pdf_part7</p>
            <p>prueba1.pdf_part8</p>
            <p>prueba1.pdf_part9</p>
            <p>prueba1.pdf_part10</p>
            <p>prueba1.pdf_part11</p>
            <p>prueba1.pdf_part12</p>
            <p>prueba1.pdf_part13</p>
            <p>prueba1.pdf_part14</p>
            <p>prueba1.pdf_part15</p>
            <p>prueba1.pdf_part16</p>
            <p>prueba1.pdf_part17</p>
            <p>prueba1.pdf_part18</p>
            <p>prueba1.pdf_part19</p>
            <p>prueba1.pdf_part20</p>
            <p>prueba1.pdf_part21</p>
            <p>prueba1.pdf_part22</p>
            <p>prueba1.pdf_part23</p>
            <p>prueba1.pdf_part24</p>
        </partes>
    </archivo>
    <archivo>
        <nombre>prueba2.pdf</nombre>
        <formato>pdf</formato>
        <totalPartes>7</totalPartes>
        <partes>
            <p>prueba2.pdf_part0</p>
            <p>prueba2.pdf_part1</p>
            <p>prueba2.pdf_part2</p>
            <p>prueba2.pdf_part3</p>
            <p>prueba2.pdf_part4</p>
            <p>prueba2.pdf_part5</p>
            <p>prueba2.pdf_part6</p>
        </partes>
    </archivo>
    <archivo>
        <nombre>prueba3.png</nombre>
        <formato>png</formato>
        <totalPartes>1</totalPartes>
        <partes>
            <p>prueba3.png_part0</p>
        </partes>
    </archivo>
    <archivo>
        <nombre>prueba4.png</nombre>
        <formato>png</formato>
        <totalPartes>1</totalPartes>
        <partes>
            <p>prueba4.png_part0</p>
        </partes>
    </archivo>
</ListaArchivos>
```

Figura 22: Captura del XML de Listado de Archivos

Sin embargo, trabajar constantemente leyendo y escribiendo el archivo XML sería muy costoso a nivel de recursos y muy poco eficiente. Por eso, después de iniciar la aplicación, se volcará la información del XML en una estructura de datos en Java que se ha llamado '*ObjetoLista*'. De esta manera la información estará mucho más accesible y la consulta y actualización de los datos será mucho más eficiente en términos de tiempo. Una vez el programa vaya a ser apagado, se volcarán los datos contenidos en el '*ObjetoLista*' de vuelta en el XML para que estén disponibles en la próxima apertura.

Para mantener '*ObjetoLista*' tan actualizado como sea posible se comprobarán los directorios locales periódicamente; también, cada vez que un archivo sea descargado de los otros nodos, el módulo de Gestión de Archivos será informado y reflejará esa información en '*ObjetoLista*'.

4.5 Módulo de Descubrimiento y Bootstrapping

Descubrimiento de la Red

La necesidad de algún mecanismo de descubrimiento de los nodos de la red es necesaria dada su naturaleza descentralizada. Desde el principio se descartó la utilización de un tracker ajeno a la red para mantenerla completamente independiente. Se opta por utilizar una IP fija para el nodo que haga las veces de Master: el desempeño será mejor en este caso sin usar el servicio de DNS dinámico, al ser el prototipo inicial de la aplicación una red con un número relativamente bajo de nodos. De esta manera, si un nodo nuevo quiere acceder a la red, sabrá a qué dirección IP referirse para iniciar el proceso de Bootstrapping.

Bootstrapping con el Nodo Master

La funcionalidad del módulo de Bootstrapping se explica en detalle en la sección 3.6 Módulo de Descubrimiento y Bootstrapping y es prácticamente igual en el desarrollo que en la idea del diseño. TomP2P, como parte de las funcionalidades relacionadas con el mantenimiento de la red y las conexiones entre nodos, cuenta con una función de Bootstrapping que va incluida en todos los Peers. De esta manera el sistema tiene un mecanismo robusto de Bootstrapping ya documentado y probado.

TomP2P no hace ninguna diferenciación entre los Peers, por lo que todos ofrecen la posibilidad de llevar a cabo el Bootstrapping con ellos. Sin embargo, en este proyecto se hace la diferenciación a nivel de nodo entre Master y nodos normales para tener un único punto de entrada a la red, aumentando la seguridad.

4.6 Módulo Central de la Aplicación

El módulo Central está en el núcleo de las operaciones, es el programa principal que desencadena el resto de acciones y por el que pasan constantemente comunicaciones entre los módulos. Su importancia es máxima especialmente al comenzar y terminar la ejecución de la aplicación.

Al inicio será el responsable de iniciar en el módulo P2P y de darle instrucciones para que inicie el Bootstrapping con el nodo Máster. Lo siguiente será iniciar la actividad del módulo de Gestión de Archivos con el análisis del XML y la actualización de los datos de los archivos en la estructura '*ObjetoLista*. Una vez actualizado, le dará instrucciones al módulo P2P para que ponga a compartir todas las partes de los archivos que se encuentren en *'ObjetoLista'.*

En este punto el nodo ya será una entidad estable que podrá interactuar con los demás componentes de la red. Durante la ejecución de la aplicación el módulo Central estará en el centro de las comunicaciones pero su actividad de verá reducida.

Cuando el usuario decida cerrar la aplicación, algunas cosas tendrán que tenerse en cuenta, no se cerrará todo de golpe. Con respecto al módulo P2P, puede darse la orden de cierre, o no; el resto de Peers se dará cuenta de que ha desaparecido y continuará sin él. Sin embargo en lo referente a los archivos, el XML se actualizará una última vez antes de cerrarse.

4.7 Interfaz Web

El Interfaz Web consta de dos partes, la gráfica y la lógica. Para la parte gráfica se ha hecho un primer diseño que pretende ser simple pero agradable a la vista. Para ello se ha desarrollado una página simple mediante HTML y CSS apoyándose en el framework de código abierto desarrollado por Twitter llamado Bootstrap[10] (aunque el nombre sea parecido, no tiene ninguna relación con el proceso de Bootstrapping de los nodos). El menú que verá el usuario tendrá los botones encima de un recuadro centrado en mitad de la página. En dicho recuadro se le presentará la información al usuario, así como las posibilidades de descarga.

La parte del Interfaz Web encargada de la lógica está escrita en JavaScript y utiliza JQuery [11], también desarrollado en JavaScript. Se ocupará de 4 funciones:

- Actualización y presentación de la información:

  Como se aprecia en las figuras 23 y 24, el panel central es donde se le presentará la información al usuario. Cada vez que se lleve a cabo una acción el cuadro se refrescará y el usuario obtendrá nueva información.

- Información de archivos del nodo asociado al Interfaz Web:

  Se solicita al nodo asociado información sobre los archivos disponibles. El nodo recibirá la petición a través del módulo HTTP y consultará al módulo de Gestión de Archivos. Con la información que reciba enviará un JSON de vuelta al navegador.

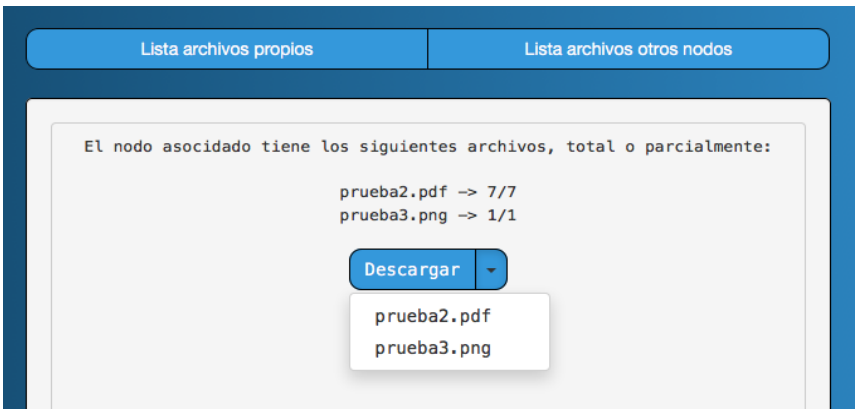


Figura 23: Información sobre archivos del nodo asociado al Interfaz Web

- Información de archivos del resto de nodos:

  De nuevo el Interfaz Web envía la petición al módulo HTTP del nodo asociado, que se comunica con el resto de nodos de la red para obtener la información necesaria. Una vez recibida, la empaqueta en un JSON y la envía de vuelta al navegador.

- Descarga selectiva de archivos del resto de nodos:

  En ambas figuras 23 y 24 se aprecia que, tanto en el listado de archivos del nodo asociado como en el listado de los demás nodos, aparece un desplegable para descargar dichos archivos.

  En el caso del nodo asociado esos archivos ya están listos para ser descargados por HTTP por el navegador. En el caso de los demás nodos, primero los descargará el nodo asociado mediante P2P para enviarlo después al navegador vía HTTP.

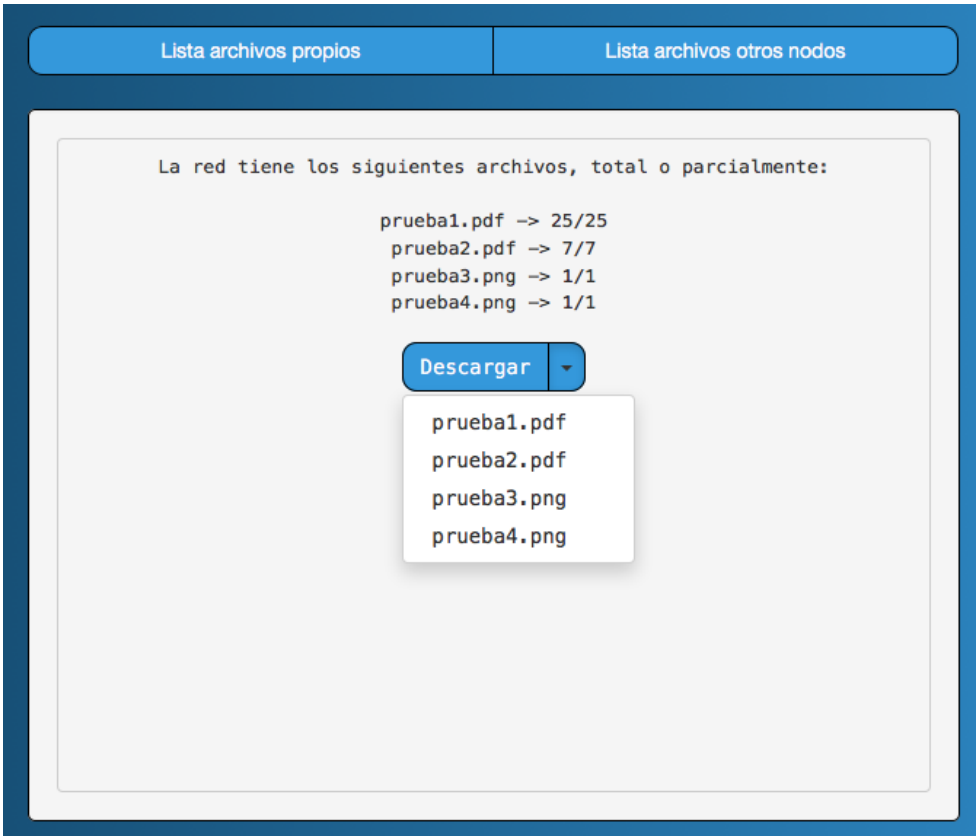


Figura 24: Información sobre archivos del resto de nodos

Capítulo V

5. Pruebas Realizadas

El proyecto se ha desarrollado a lo largo de varios meses, por lo que ha habido diferentes estados en los que se ha encontrado y diferentes pruebas que ha sido necesario llevar a cabo a lo largo de la vida del mismo para continuar con su desarrollo.

Al haberse planteado el diseño de la aplicación de forma lo más modular posible, las pruebas de las diferentes partes del proyecto se han hecho de manera independiente primero, y después de manera conjunta. Todas las pruebas que dan resultados positivos en el programa final los dieron también en las pruebas iniciales independientes, por lo que no se detallan.

Las pruebas que se describen a continuación evalúan principalmente la posibilidad de llevar a cabo determinadas acciones; esto es debido a que el proyecto no trataba de evaluar el desempeño de un programa ya existente, si no que más bien era una prueba de concepto con diferentes puntos de control.

5.1 Pruebas en los Módulos

5.1.1 Módulo P2P

Como se explica en la sección 3.3 Módulo P2P, inicialmente se optó por una implementación P2P llamada Ttorrent con la que se llegó a una primera versión funcional, pero que necesitaba de terceras partes para llevar a cabo el descubrimiento de la red.

Ttorrent

Prueba realizada	Evaluación de la prueba	Comentarios
Interconexión de varios nodos	Satisfactoria	
Descubrimiento de la red sin necesidad de terceras partes	Insatisfactoria	Necesidad de un tracker en un servidor para el descubrimiento

Con la sustitución de Ttorrent por TomP2P quedó solucionado este problema y se pudo continuar con las pruebas necesarias.

TomP2P

Prueba realizada	Evaluación de la prueba	Comentarios
Interconexión de varios nodos	Satisfactoria	
Descubrimiento de la red sin necesidad de terceras partes	Satisfactoria	
Auto mantenimiento de la red de forma "indefinida"	Satisfactoria	
Anuncio de archivos disponibles para descarga en un Peer	Satisfactoria	
Descarga de archivos anunciados previamente por un Peer	Satisfactoria	

5.1.2 Módulo HTTP

Con respecto al módulo HTTP las pruebas fueron todas satisfactorias, y solamente hizo falta cambiar una función para hacerla más eficiente.

Prueba realizada	Evaluación de la prueba	Comentarios
Comunicarse con otros nodos y con el Interfaz Web indistintamente	Satisfactoria	
Servir los archivos necesarios para el Interfaz al navegador	Satisfactoria	
Descargar y combinar listados de archivos en formato XML disponibles en otros nodos	Satisfactoria	Aunque se implementó, está funcionalidad era ineficiente por enviarse mucha más información de la necesaria

Prueba realizada	Evaluación de la prueba	Comentarios
Descargar y combinar listados de archivos en formato JSON disponibles en otros nodos	Satisfactoria	Corrige la ineficiencia de la prueba anterior empaquetando solamente la información necesaria
Enviar archivos seleccionados por el usuario para descarga a través del Interfaz Web	Satisfactoria	

5.1.3 Módulo de Gestión de Archivos

Aunque fue de los módulos que más tiempo llevó y cuyas funciones son las más extensas, todas las pruebas fueron satisfactorias al final.

Prueba realizada	Evaluación de la prueba	Comentarios
División de archivos en partes de igual tamaño	Satisfactoria	
Volver a unir las partes para formar el archivo original, incluso si las partes vienen de fuentes distintas	Satisfactoria	
Inspeccionar los directorios para saber qué archivos y qué partes hay disponibles	Satisfactoria	
Sacar la información del XML (mediante XPath) y cotejarla con *'ObjetoLista'*	Satisfactoria	
Actualizar correctamente de manera periódica tanto *'ObjetoLista'* como el XML	Satisfactoria	

5.1.4 Módulo de Descubrimiento & Bootstrapping

Como se menciona en las pruebas del módulo P2P, se utiliza TomP2P para las funciones de Bootstrapping de este módulo. Para el descubrimiento de la red no se consiguió llegar a implementar con un servicio de DNS dinámico.

Prueba realizada	Evaluación de la prueba	Comentarios
Descubrimiento con IP fija del nodo Master	Satisfactoria	
Bootstrapping de un nodo nuevo, e inclusión en la red	Satisfactoria	

5.1.5 Interfaz Web

Tanto la parte gráfica como la lógica del Interfaz Web funcionaron bien desde el principio y no hubo mayores complicaciones.

Prueba realizada	Evaluación de la prueba	Comentarios
Diseño e implementación de un cuadro central para presentar la información al usuario	Satisfactoria	
Funcionalidades en forma de botón capaces de mandar peticiones de información al nodo asociado	Satisfactoria	
Sub-menús de descarga del archivo seleccionado por el usuario	Satisfactoria	

Capítulo VI

6. Planificación y Presupuesto

6.1 Planificación

Dada la complejidad del proyecto, se organizaron las tareas y los tiempos desde un primer momento. A continuación se muestran tanto la lista de tareas y qué componente del proyecto debía realizarlas como el diagrama de Gantt con los tiempos de cada una.

	Task Name	Duration	Start Date	End Date	Predecessors	Assigned To
1	**Fase Inicial y Mantenimiento**	**156d**	**21/09/15**	**25/04/16**		**Jefe de Proyecto**
2	Toma de requisitos	3d	21/09/15	23/09/15		Jefe de Proyecto
3	Planificación	3d	24/09/15	28/09/15	2	Jefe de Proyecto
4	Mantenimiento planificación	150d	29/09/15	25/04/16	3	Jefe de Proyecto
5	Reunión 1	1d	05/10/15	05/10/15		Jefe de Proyecto
6	Reunión 2	1d	02/11/15	02/11/15		Jefe de Proyecto
7	Reunión 3	1d	07/12/15	07/12/15		Jefe de Proyecto
8	Reunión 4	1d	11/01/16	11/01/16		Jefe de Proyecto
9	Reunión 5	1d	01/02/16	01/02/16		Jefe de Proyecto
10	Reunión 6	1d	07/03/16	07/03/16		Jefe de Proyecto
11	Reunión 7	1d	28/03/16	28/03/16		Jefe de Proyecto
12	Reunión 8	1d	25/04/16	25/04/16		Jefe de Proyecto
13	**Fase de Diseño**	**40d**	**06/10/15**	**30/11/15**		**Analista**
14	Estudio Estado del Arte	15d	06/10/15	26/10/15	5	Analista
15	Diseño Estructura General	5d	27/10/15	02/11/15	14	Analista
16	Diseño Módulos	20d	03/11/15	30/11/15	15	Analista
17	**Fase de Implementación**	**92d**	**01/12/15**	**06/04/16**		**Desarrollador**
18	Instalación y Preparación Herramientas	2d	01/12/15	02/12/15	13	Desarrollador
19	Pruebas	92d	01/12/15	06/04/16	13	Analista
20	Implementación Estructura Nodo	45d	03/12/15	03/02/16	18	Desarrollador
21	Implementación Red	30d	04/02/16	16/03/16	20	Desarrollador
22	Implementación Interfaz Web	15d	17/03/16	06/04/16	21	Desarrollador
23	**Fase de Documentación**	**156d**	**21/09/15**	**25/04/16**		**Analista**

Figura 25: Lista de tareas

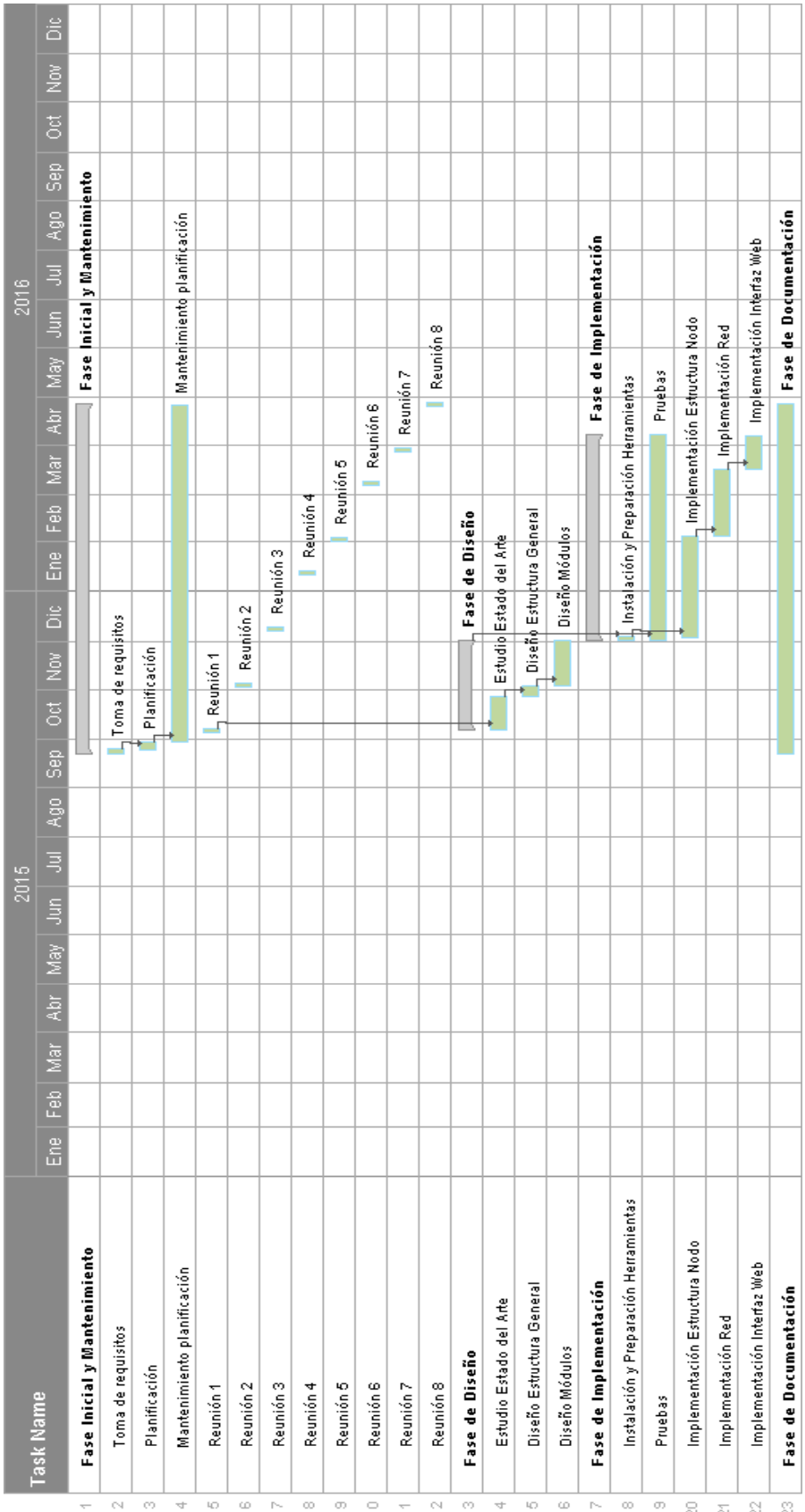


Figura 26: Diagrama de Gantt

6.2 Presupuesto

6.2.1 Coste de Personal

Aunque el trabajo no se ha distribuido de igual manera a lo largo de todo el desarrollo del proyecto, se fija una media de 3,5 horas diarias de trabajo. Por tanto, podemos presentar las horas en función de quién realiza las tareas o en función de la fase llevada a cabo; en ambos casos el total de tiempo dedicado asciende a 521,6 horas.

En función de la fase del proyecto

Fase Inicial y Mantenimiento:

$$6\ días \cdot 3{,}5\ horas + 150\ días \cdot 0{,}1\ horas + 8\ días \cdot 1\ hora = 44\ horas$$

Fase de Diseño:

$$40\ días \cdot 3{,}5\ horas = 140\ horas$$

Fase de Implementación:

$$92\ días \cdot 3{,}3\ horas + 92\ días \cdot 0{,}2\ horas = 322\ horas$$

Fase de Documentación:

$$156\ días \cdot 0{,}1\ horas = 15{,}6\ horas$$

En función de quién realiza las tareas

Jefe de Proyecto:

$$6\ días \cdot 3{,}5\ horas + 150\ días \cdot 0{,}1\ horas + 8\ días \cdot 1\ hora = 44\ horas$$

Analista:

$$40\ días \cdot 3{,}5\ horas + 92\ días \cdot 0{,}2\ horas + 156\ días \cdot 0{,}1\ horas = 174\ horas$$

Desarrollador:

$$92 \cdot 3{,}3 = 303{,}6\ horas$$

Estos últimos datos serán los interesantes para hacer el cálculo de los salarios a lo largo del proyecto, considerando que la dedicación de un hombre al mes es de 80 horas. Se presenta dicha información en la siguiente tabla:

Puesto	Dedicación Mensual	Coste Mensual *	Coste Total
Jefe de Proyecto	0,55	4.000 (50€/hora)	2.200 €
Analista	2,175	2.000 (25€/hora)	4.350 €
Desarrollador	3,795	1.600 (20€/hora)	6.072 €
			12.622€

* tomando como referencia que un hombre al mes son 80 horas

6.2.2 Coste de Hardware & Software

A pesar de que el hardware utilizado es propiedad del autor, se calcula la amortización de dichos elementos. El software no tiene coste alguno ya que se ha desarrollado la mayor parte y el resto de lo que se ha utilizado es software libre.

Ítem	Coste	% uso dedicado	Dedicación Meses	Período Depreciación	Coste Imputable
Macbook Pro I5 2500 4Gb RAM 500Gb	1000€	100	7	60	126€
					126€

Cálculo de la amortización: $\frac{A}{B} * C * D$

A -> número de meses desde la fecha de facturación en que el equipo es utilizado
B -> período de depreciación (60 meses)
C -> coste del equipo (sin IVA)
D -> porcentaje del uso que se dedica al proyecto (habitualmente 100%)

6.2.3 Resumen de los Costes

Finalmente se reúnen todos los costes del proyecto en la siguiente tabla, incluyendo un 20% de costes indirectos y un 21% de Impuesto sobre el Valor Añadido.

Descripción	Coste
Personal	12.622€
Hardware	126€
Software	0€
Subcontratación de tareas	0€
Costes Indirectos (20%)	2549,6€
Total	15.297,60€
Total con IVA	**18.510,10€**

6.2.4 Plantilla del Presupuesto

PRESUPUESTO DEL PROYECTO

1- Autor:
Guillermo Fernández Gorostidi

2 - Departamento:
Ingeniería Telemática

3 - Descripción del Proyecto:

- Titulo	Almacenamiento y Sincronización de Ficheros Alternativo a La Nube Mediante P2P e Interfaz Web
- Duración (meses)	7
- Tasa de costes indirectos	20%

4 - Presupuesto Total del Proyecto (valores en Euros):
18.510,10 €

5 -Desglose Presupuestario (costes directos):

PERSONAL

Apellidos y Nombre	NIF	Categoría	Dedicación (hombres/mes) *	Coste hombre/mes **	Coste (Euros)	Firma de Conformidad
Guillermo Fernández Gorostid		Jefe de Proyecto	0,55	4.000,00 €	2.200,00 €	
Guillermo Fernández Gorostid		Analista	2,175	2.000,00 €	4.350,00 €	
Guillermo Fernández Gorostid		Desarrollador	3,795	1.600,00 €	6.072,00 €	
				Total	**12.622,00 €**	

* 1 Hombre/mes = 80 horas
** Jefe de Proyecto 50€/hora; Analista 25€/hora; Desarrollador 20€/hora

EQUIPOS

Descripción	Coste (Euros)	% Uso dedicado al proyecto	Dedicación (meses)	Periódo Depreciación	Coste Imputable ***
Macbook Pro I5 2500 4Gb RAM 500 Gb	1.000,00 €	100	7	60	126,00 €
				Total	**126,00 €**

*** Fórmula de cálculo de la amortización (A·C·D)/B
A -> número de meses desde la fecha de facturación en que el equipo es utilizado
B -> período de depreciación (60 meses)
C -> coste del equipo (sin IVA)
D -> porcentaje del uso que se dedica al proyecto (habitualmente 100%)

6 - Resumen de Costes:

Descripción	Presupuesto Costes Totales
Personal	12.622,00 €
Amortización	126,00 €
Subcontratación tareas	0,00 €
Otros costes directos	0,00 €
Costes Indirectos	2.549,60 €
Total	**15.297,60 €**
Total con IVA (21%)	**18.510,10 €**

Capítulo VII

7. Conclusions & Future Work

7.1 Conclusions

This project was meant to provide the file exchange service that we are so used to get from The Cloud, only in a safer and distributed way. The project has implemented a first approach to an overlay network of nodes that does not depend on third parties for any of their activities, allowing the nodes to exchange files from anywhere in the Internet, by means of a web interface. Each of these nodes may come and go from the network and its performance will not be affected. As a proof of concept, it has been actually demonstrated that such an idea is viable, as all the main objectives have been fulfilled.

The network has been structured putting the node in the nucleus. The node's structure has been designed and implemented in a highly modularized way, so that the different parts work independently but are able to interact efficiently. It was decided to do it this way so that the modules would be easily replaced, if there were improvements to be made later on, without affecting the performance of the rest of the node entity. This was actually the case when a first Peer-to-Peer client was implemented (Ttorrent), but was later decided to change it for another one (TomP2P) as the first one did not fulfill the network discovery needs.

From the beginning, the network was always meant to be as independent from third parties as possible, but during the implementation of Ttorrent in the P2P module it was made clear that a tracker was needed in a third party server in order to perform the network discovery functions. As this was not aligned with the needs of the project, Ttorrent was replaced with TomP2P, which offered a discovery functionality independent of anything outside the network. This process is called Bootstrapping and will have the need of two different kinds of nodes: Master node and regular node, although the node structure has been implemented in a way that any node can play both roles if needed. There will only be one Master node simultaneously in the network, and it will be in charge of receiving the entry requests and perform the Bootstrapping

process in which the incoming node will receive the necessary information about the topology of the overlay network.

Although with the P2P connection between the nodes the network was already given a structure and it was possible to exchange files between the devices, in order to achieve all the functionalities offered by a Cloud based file exchange service, the files needed to be accessible from any device on the Internet. Analogously to the Cloud services, this issue was solved by means of a web interface accessible from any device that has a web browser. Hence, the node incorporates another functionality, working as a web server to the interface: the node will serve the webpage of the interface and will provide the necessary tools for the user to manage the files from any device. This interface will be the user's window to the application, connecting to its associated node and letting the user manage both the files in that specific node and in the rest of the network.

Regarding the information exchanged, there will be two types of data travelling between the nodes: messages regarding the files available, and the actual files that are exchanged. Although both kinds of data exchange could have happened through the same module of the node, it was separated so that the HTTP module would take care of the information about the files, while the P2P module exchanges the parts of the files. This is not a trivial decision, two advantages arise from this approach: first, the P2P network will be much more efficient when exchanging large amounts of data, as will happen when files are sent; and secondly, in order for the web interface to be able to communicate with the nodes as fluently as they communicate with each other, an API REST was implemented in the HTTP module. The web interface is not a node like the others, so this last reason is especially important to unite communications and to eliminate the need of translating the web messages into something the nodes can understand.

As for the file management, the files will be divided in parts of fixed size for sharing, what will make the download from several nodes at once more efficient. As the upload rate of the sending nodes is usually lower than the download rate of the receiving one, if data is downloaded simultaneously from different sources there will happen no bottleneck in those sources, but the download rate of the receiving one will be optimized. The information about the files in each node will be managed through data structures especially created for this project: one of them will be an XML file that will keep the information when the application is closed, and the other one fill be a structure that will be more dynamic and accessible during runtime.

Overall, the initial objectives and specifications have been fulfilled; the main functionalities of a Cloud based file exchange service have been implemented by means of a Peer-to-Peer network, complemented with a way of managing the files from any point of the Internet. Nevertheless,

had there been more time available, improvements and enhancements would be possible. More than error fixing, the improvements would be focused in expanding the capacity and functionality of the application. Some of the ideas that could be interesting range from making it a multi-device application to improving the security of the communications and the web interface.

7.2 Future Work

During the development of this project a lot has been learned about P2P networks and file management. Nevertheless, had there been more time available more functionalities would have been implemented to complete de existing version. If the project was to continue, the following aspects would be interesting to include:

- Make it a multi-device application:

  As explained in the 4.2 Módulo P2P section, TomP2P was selected, among other reasons, for having a working implementation on Android that would enable the nodes to also be run on mobile devices and tablets. As the rest of the modules are written in Java and are quite lightweight, the application could easily be migrated to these devices.

  If this was to happen, the politics module that was described in 3.8 Módulo de Política section would be needed, which has not really been the case until now since the development has been done for larger "unconstrained" devices. Aspects such as the battery of the small device or the fact that it may be connected to the Internet through mobile data networks may have to be taken into account in order for the normal execution of the application to not be a problem.

- Security enhancements:

  Especially regarding the Web Interface, which is probably the most vulnerable point of the network. Authentication methods such as the ones described in the design would need to be implemented before the application could be published for use. Although the overall user experience would change little or nothing by adding a user-password entry to the Web Interface service, it would be an important security measure.

  Also, the access of new nodes to the network would need to be controlled by the Master node, who would check if someone in the existing network invited the incoming node. This would, however,

require quite some time to develop and more research into these kind of security functionalities.

- Easier access to the network:

  Although the current solution with fixed IP address of the Master node is fully functional, it works efficiently as the networks tested were not too big. It would be advisable to be prepared for a wider number of composing nodes in the network, for what a dynamic DNS approach would work better.

- More functionalities available to the user:

  As for a first version of a file exchange service, this project fulfills the basic needs of a user, but more functionalities would be welcomed and would complete the experience. These could, for example, include things such as the deletion of files of the network from the Web Interface or an automatic backup of certain important files selected by the user into the rest of the devices.

Capítulo VIII

8. Entorno Socioeconómico y Legal

Desde su creación y posterior popularización, los programas basados en redes P2P han sido siempre perseguidos por la polémica. Hacia 2006 aproximadamente el 60% del tráfico total de Internet era debido a las redes P2P [12], por lo que su relación con la legalidad adquiere bastante importancia.

Aunque dichas redes sirvan, en el caso del intercambio de archivos, para hacerlo de manera neutral y sin relación con lo que se comparte en ellas, en ocasiones se han utilizado para compartir archivos de manera ilícita. El debate se centra esencialmente en el intercambio de archivos que pudieran estas sometidos a derechos de propiedad intelectual, por lo que juegan un papel importante tanto las asociaciones de protección de los derechos de autor como los grupos a favor del uso compartido de la propiedad intelectual.

A pesar de ser un tema cada vez más relevante, no existe en España una Ley que regule su uso ni fije directrices o penalizaciones. Hay, sin embargo, un histórico de sentencias judiciales contra los creadores de programas P2P y los web masters de páginas de listados de enlaces a redes P2P. Estas sentencias han dado la razón en la inmensa mayoría de los casos a los demandados, siendo por ejemplo un caso muy sonado el de la SGAE contra Jesús Guerra en 2010 por una página de enlaces a contenidos P2P [13] y el de varias multinacionales discográficas como Sony o Warner Music contra Pablo Soto en 2014 por el desarrollo de software P2P [14].

En febrero de 2011 el PSOE inició el desarrollo de la denominada Ley Sinde, que heredó el PP en la legislatura siguiente pasando a denominarse Ley Sinde-Wert, llegando a aprobarla dos años y medio más tarde. Esta ley buscaba penalizar a las páginas de listados de contenidos. Aunque ha tenido un efecto limitado, algunas páginas han sido afectadas por la ley, principalmente las de listados de contenidos para descarga directa, aumentando más aún el uso de las redes P2P.

De manera más reciente, y aún sin una ley que afecte de manera directa al uso de las redes P2P, la Fiscalía General del Estado emitió a

finales de 2015 una circular que indica procedimientos y criterios a seguir en litigios relacionados con el tema [15]. De este documento se desprenden los siguientes puntos importantes:

- No hay delito penal en la descarga de archivos mediante redes P2P siempre que dichos archivos sean legales, es decir, no infrinjan otras leyes como podría ocurrir si se compartiera, por ejemplo, una película grabada directamente con una cámara de la pantalla de un cine.

- Con respecto a los archivos protegidos por derechos de autor, estos podrían compartirse sin el permiso ni el conocimiento de dicho autor siempre y cuando no se obtuviera ningún beneficio económico a cambio, ya sea directo o indirecto.

Estas indicaciones de la Fiscalía General del Estado vienen a reforzar la idea ya implantada en la comunidad P2P de que el intercambio de archivos a través de estas redes está más en la línea de compartir contenido como pudiera ocurrir en el mundo real, más que como un intercambio ilícito de contenido de pago. Es decir, sería como recibir una copia de un disco de alguien que lo ha comprado, en vez de como robar ese disco en una tienda.

En lo tocante a este proyecto, todo lo anteriormente explicado aplica incluso aunque sea una red F2F. El hecho de que la red esté constituida por un grupo de gente que se conoce y no se compartan los archivos con toda la red de Internet afianza incluso más la idea de que no se hará un uso comercial de archivos que tengan derechos de autor.

Apéndice A

A. Summary

A.1 Introduction

Traditionally, online services have ran following a client-server fashion. Every time we make a search on a search engine, we download a webpage or use any other service, we are normally communicating in our client role with a server that provides the contents and services.

Nevertheless, as the Internet has been growing bigger and the number of users on It has increased, a new model different to the traditional one has gained momentum: Peer-to-Peer networks. This architecture shares resources among all the nodes that conform the network, increasing considerably the capacity of each of them. In it, no node plays the client or the server role, but rather all the nodes play both roles simultaneously. The distributed nature of the network also brings security advantages as even when a node is compromised, the rest can still work normally; there are no single points of failure as in the client-server model.

More and more applications are continuously published and commercialized based on this structure rather than on the traditional one, reaching already big companies' services. To give a notorious example, the well-known communication application from Microsoft, Skype, is based on this P2P structure.

Still, what P2P networks are most popular for are the file sharing applications, mainly the ones that use the BitTorrent protocol. Year after year the bandwidth of the Internet dedicated to the exchange of files through these means has only but increased. This has happened without the need of any central server for controlling the flow of data, the files or any such things. As the users of the Internet become more aware of the advantages of sharing resources among several nodes, the popularity and efficiency is only likely to keep increasing, as this model gets even better when scaled.

On the client-server side, a term that we are used to reading and hearing about a lot is The Cloud: an uncertain number of servers located in arbitrary locations that allow us to use their resources. Especially for file storage, we use these services on a daily basis, mainly due to the convenience of having our files available in whichever device never mind where we are at the moment.

Despite the advantages of such a service, uploading files to a server of which we have no control or knowledge whatsoever does have some serious drawbacks. The situation, although it is not realized most of the times, is that we are uploading our personal files to a server to which we do not know who is able to access the files, which is its location or how these are actually stored.

This project has been designed to try to solve these serious security and privacy flaws using a P2P approach. More precisely, the model that will be used is a Friend-to-Friend approach, a more specific type of P2P network. In the F2F model, unlike general P2P applications, not every user can have access to the files; rather only some specific nodes, which are trusted, will be able to exchange them.

This will result in a situation similar to that of a shared folder in which only the users that are part of the folder are able to see and exchange files through it. In this way the security and privacy issues will be solved, as the files will be located in trusted devices and they will be as securely stored as the user wants them to. On top of all this, the advantages of the P2P architecture will still hold, and an overlay logical network will be built independently over the physical one.

As an extra measure for enhanced performance, inspired in the BitTorrent protocol, the files will be divided into smaller parts and stored separately. This means that different nodes may have different parts of a file. Other nodes willing to download that file will be able to distribute its download among the nodes that have parts of it, getting from each of them different parts of it. This will accelerate the download and reduce the bandwidth used by the sending nodes.

Finally, in order to offer the exact same functionalities as a cloud-based file storage and sharing service, there will be a web interface that will be able to connect to the rest of the components of the subnetwork.

A.2 Structure and Objectives

A.2.1 General Structure

The application, as a F2F network, will need to have certain components that will provide its structure. Both the elements that will compose the network and the way in which they will communicate to each other need to be designed.

The first and most basic component over which everything else will be built is the node. The node entity will wrap all the rest of the components together, building a self-contained connectable entity that is able to store files in the device and makes use of a data structure with meta information about those files. In other words, the node will be in the core of the model, and it will be through it that the communications will happen and where the information from each user will be stored.

There will be a structure created in order for each node to be aware of what parts of what files are under their control. Two data structures will be created to fit our needs: the first one will be an XML that will play its most important part when the application is not running, storing the meta information about the files until the program is restarted; the second one will be a data structure in Java developed for this particular project, and will be used during runtime of the program as it will be more dynamic and accessible than the XML.

In order to interconnect the node elements, there will be a communication protocol defined that all of them will understand and use. This protocol will be designed through a REST API so that connection to the nodes from a web interface will be possible without any further translation of the protocol messages.

The last part of the network will be the web interface. It will be the way in which the user is able to interact with the application by being presented some information about the files and taking some actions as a response. The web interface will not be a regular node like the rest, but rather it will be a window from a web browser into the P2P network. Each web interface will be associated to a node and the communications and downloads from the rest of the nodes will happen through that associated node.

A.2.2 Interconnection and Discovery

Interconnection among the nodes will be implemented in a F2F fashion. This will have certain implications that may not seem obvious at first glance.

When a node that is still not part of the network wants to become part of it, there will be some information missing about how the network is built and what nodes form it already. This node will fail to enter the network unless aware of the specific discovery protocols, so another node which is already part of the network needs to provide this information.

Provided that the new node is legitimately trying to enter the network, a process called Bootstrapping will come to play its part. By means of Bootstrapping to an already connected node, the new one will gather all this information that it is missing to be able to communicate with the rest. Once the new node has correctly bootstrapped with one of the conforming nodes, it will become an equal member of the network, having no more and no less functionalities than the rest, and becoming itself a potential bootstrapping peer for new nodes wanting to enter.

The discovery of the network would become extremely complicated if the Bootstrapping could be done with any of the nodes, hence a differentiation between two kinds of nodes is made: Master node and regular node. All the nodes will have the potential of being the Master node, which will not have more functionalities but rather more responsibilities, but there will only be on Master node at a time. This master node will be the one through which the Bootstrapping will take place.

Also, in order to avoid any need of third party services, such as a tracker in a server, the Master node should be easily accessible. This can be managed in two ways: making the Master node have a fixed IP that will be public, which will be an efficient solution if the network is not too big; the second solution will be the use of a dynamic DNS service which will update the location of the Master node constantly, which will be more efficient if the number of nodes in the network gets high enough.

A.2.3 Communication

In an already stable network, that is, once all the nodes that will be conforming the network are already connected, information and file exchange can take place. There will be differentiation in the case of information exchange (through the HTTP module) as opposed to file exchange (through the P2P module).

The information about the files stored will be exchanged when someone in the network wishes to know what files other nodes have, for example to fill the list of possible downloads in the web interface.

The relevant information will be what parts of the files each node has. This information will be asked for to each node separately, and the information will then be processed back at the node that asked for it. This means that the node willing to obtain the information will send a message to each one of the other nodes, similar to a broadcast message, and each node will answer back with a similarly structured message containing the information required.

The file exchange will happen once the information about the parts has been exchanged, for example if the user of the web interface decides to download a specific file.

In any of the cases, the receiving node will send download messages for specific parts of a file to certain nodes. Those messages will be different for each node depending on the parts that it has. The balancing of what parts will be asked for to what nodes will be decided by the node previous to sending the download messages and this decision will be taken based on criteria that will optimize the download time and bandwidth of the nodes.

A.2.4 File Exchange

Initially two possibilities were considered that would imply file exchange, one in which files were automatically synchronized between the nodes, and another one through selective download of files by the user. The first case was then discarded as it was considered that the whole network would already work as a sort of shared folder and that the point

was that the user was able to access the files from any device, rather than making a copy of each file in each device.

Hence, on the selective download scenario, the web interface played its part. The user will be able to ask for the files already present in the associated node/device and for the files available in the rest of the network. Once the information is presented, he will have the possibility of downloading the files from the associated node or from the rest of the nodes.

This will not make a huge difference, other than the fact that if the file downloaded is already available in the associated node the transfer will only need an HTTP connection. If the file is to be downloaded from the network, the parts will be requested to the other nodes via P2P by the associated node, which will reassemble the parts and send it through to the web interface again by HTTP.

A.3 Results

As this project was more a proof of concept, rather than the performance evaluation of an already implemented idea, the tests undertaken were to give satisfactory or unsatisfactory results, rather than quantitative ones.

After the development of a first version, most of the initial objectives have been fulfilled and a working implementation is available. Since the initial idea, some things needed to be changed:

For the P2P module, Ttorrent was initially considered and even a working version was even achieved. Unfortunately, later on it was realized that, although the rest of the functionalities could be implemented with this software, the discovery feature needed a tracker to happen. This was a point that wanted to be avoided since the design face, so a new P2P client was needed. After some research, TomP2P was found and it was checked that it fulfilled also the Bootstrapping needs of the project.

Another aspect that needed to be revised was how the meta information about the files was exchanged. At first it was done by sending the whole XML available in each node, but this was highly inefficient as it's processing was way slower and difficult, especially in the web browser. Taking advantage of the fact that the HTTP module was implemented with a Jersey server that supported JSON messages, this structure was used instead. JSON messages are the most used data structure across web browsers and put together information in a more efficient way.

A lot has been learned on various areas during the development of this project, being the first relatively big academic paper carried out by the author. The obstacles, timing issues and research needs that have arisen during the development of the project have been a quite enriching experience.

A.4 Conclusions

The implementation of this project has proved that a P2P based approach to a file exchange application is plausible and can provide the same functionalities that The Cloud gives with it's client-server approach. No third parties were needed to offer access to the files of the rest of the devices, even from the web interface. Also, given the distributed nature of the network, no points of failure are present and the network can continue if any of the nodes is shut down.

Most of the criteria in the design were implemented. Nevertheless, a lot of space for improvement is still present and more functionalities could be added to this first working version; had there been more time, they would have been implemented.

It would have been really interesting to make the application a multi-device one, and in fact TomP2P was chosen because of having a working implementation for Android that would enable the nodes to also be run on mobile devices and tablets. This would also mean the further implementation of the politics module, that has not been needed until now as the devices used were larger and relatively unconstrained.

Another important improvement that could be carried out would be to enhance security, especially regarding the authentication. The web interface should only be accessible after going trough a controlled entry granted only by the correct credentials. Also, authentication should be stronger for new incoming nodes entering the network through the Bootstrapping feature. Until these authentication characteristics were completely robust, the application could not be published for a wider use.

Apéndice B

B. Introducción

B.1 Motivación del Proyecto

Las redes a las que nos referimos como tradicionales, es decir las basadas en una estructura de cliente-servidor, son a día de hoy las más utilizadas para la mayoría de los servicios en Internet. Estas redes tradicionales tienen sus ventajas y sus desventajas, pero empiezan a perder terreno para dejar paso a redes Peer-to-Peer.

Este fenómeno de migración hacia las redes P2P es algo que podemos ver ya en muchos servicios. La razón para que esto este ocurriendo es que el rendimiento de las redes en las que se comparten los recursos es mayor. Además, este rendimiento aumenta con el número de nodos que se unen a dicha red; la escalabilidad de las redes P2P hacen de su uso un hito inevitable en la historia de Internet. De esta manera, hay siempre una posible versión basada en P2P de cualquier servicio que se ofrezca con la estructura tradicional, y los que no están ya desarrollados van viendo la luz gradualmente.

En el caso de este proyecto, la idea está centrada en el almacenamiento y la sincronización de archivos. El almacenamiento en la nube es algo a lo que estamos no ya acostumbrados, si no con lo que contamos para innumerables actividades de nuestro día a día. Ya sea como copia de seguridad de los archivos, para tenerlos disponibles en el trabajo o la Universidad, o con cualquier otro fin, cada vez más se tiende a no depender exclusivamente de los recursos locales de nuestro dispositivo.

A pesar de todas las ventajas que nos ofrecen estos servicios en la nube, se pasan por alto dos de los aspectos más importantes, si no los más importantes de todos, la seguridad y la privacidad de los archivos almacenados. Al subir los archivos a un servidor de un tercero, perdemos por completo el control sobre estos archivos; ya no sabemos dónde se encuentran, cómo se almacenan ni quién tiene realmente acceso a ellos. Lo que en el almacenamiento local es algo básico y completamente bajo control por el usuario, en los servicios en la nube se puede ver comprometido con demasiada facilidad.

Surge así la idea de desarrollar un programa de sincronización de archivos entre varios dispositivos o usuarios utilizando una red P2P. Hay ya disponibles en Internet un sinfín de programas que dan acceso a redes P2P para descargar archivos y compartirlos con otros usuarios, incluso sin saber realmente quienes son estos usuarios. Sin embargo, este proyecto se centra en compartir y sincronizar archivos con usuarios conocidos y autenticados, más en la línea de las redes Friend-to-Friend.

El problema de la seguridad quedará resuelto en tanto en cuanto el usuario tenga un entorno local seguro. Es decir, todos los archivos que estén almacenados y controlados por la aplicación lo estarán de forma tan segura como el usuario de la máquina los asegure; el control de la seguridad vuelve a estar en sus manos.

De la misma manera, todos los archivos viajarán cifrados de un usuario a otro y solamente los usuarios que formen parte de la comunicación serán capaces de leer los archivos que se compartan. Si un atacante interceptara en algún router intermedio alguno de los paquetes pertenecientes a un archivo, no sabría qué es lo que se está compartiendo.

Ya hay algunas versiones iniciales de programas parecidos publicadas en la red, pero no gozan de la popularidad que en un principio cabría esperar teniendo estas ventajas sobre el almacenamiento en la nube. Esto es debido a que muchas veces el usuario prefiere la comodidad antes que la seguridad, por lo que prefiere poder acceder a sus archivos desde cualquier dispositivo y no solo desde aquellos que formen parte de la carpeta compartida. Esto se consigue en los sistemas en la nube mediante conexiones vía web al servidor de almacenamiento.

Esta carencia buscará ser solucionada en este proyecto mediante el desarrollo de un interfaz web al que se accederá mediante unas credenciales y que se comunicará con los dispositivos que formen parte de la carpeta compartida, teniendo acceso así a los archivos desde cualquier dispositivo con conexión a Internet.

B.2 Objetivos

Este proyecto tendrá como objetivo principal crear un servicio de intercambio y sincronización de archivos sobre una red overlay F2F que sea accesible a través de un interfaz web que se conecte directamente con los nodos. Para ello será necesaria la implementación de los siguientes elementos:

- Se creará una estructura de nodos con idénticas funcionalidades, que serán capaces de comunicarse entre ellos y de compartir información y archivos de manera privada a través de una red F2F superpuesta a la red física.

- El intercambio de estos archivos se hará en base a cierta información guardada en estructuras diseñadas específicamente para este proyecto. Estas estructuras estarán presentes en cada nodo y guardarán información acerca de la disponibilidad de cada parte de cada archivo en sendos nodos.

- Para recorrer estas estructuras se desarrollará un método de búsqueda óptimo en base a un identificador único de cada archivo.

- Se desarrollará también un protocolo de comunicación entre los nodos para dar respuesta a todas las posibles acciones que se puedan llevar a cabo en la red.

- Este protocolo de comunicación deberá poder ser utilizado también por el dispositivo desde el que se acceda a través del interfaz web, por lo que se definirá sobre una API REST.

- Será necesario desarrollar un interfaz web que sea atractivo para el usuario y con el que se pueda hacer uso de todas las funcionalidades del programa.

- Finalmente, para hacer frente a los aspectos de seguridad se desarrollará un sistema de autenticación de los nodos y de los usuarios que se conecten a través del interfaz web.

B.3 Contenidos del Documento

Esta memoria constará de los siguientes capítulos:

- Capítulo 1 – Introducción
  Contiene una introducción al tema de la sincronización en la nube y la alternativa en redes P2P, constando de una breve motivación del proyecto, una declaración de objetivos y la presente lista de contenidos.

- Capítulo 2 – Estado del arte
  Expone el estado del arte de las redes P2P/F2F y da algunos ejemplos y explicaciones de aplicaciones desarrolladas basadas en ello.

- Capítulo 3 – Diseño
  Describe el proceso de diseño de una red estructurada sobre F2F de sincronización de archivos. Más concretamente, se explicarán los diferentes módulos que conforman el nodo, la entidad básica del proyecto.

- Capítulo 4 – Desarrollo
  Explica el desarrollo de los diferentes componentes de la aplicación; los nodos y la comunicación entre ellos, la estructura de la red y el acceso a través de la web.

- Capítulo 5 – Pruebas
  Contiene los criterios de evaluación de los objetivos principales, así como las pruebas realizadas y su análisis.

- Capítulo 6 – Planificación y presupuesto
  Describe las diferentes etapas por las que ha pasado el proyecto en base a los tiempos y las actividades llevadas a cabo, así como un presupuesto aproximado.

- Capítulo 7 – Conclusiones y trabajo futuro
  Cierra el proyecto exponiendo las conclusiones obtenidas y da unas posibles ideas para un trabajo futuro en la aplicación.

- Capítulo 8 – Entorno socioeconómico y legal
  Analiza brevemente el entorno legal en el que se encuentran este tipo de redes y aplicaciones.

Apéndice C

C. Conclusiones y Trabajo Futuro

C.1 Conclusiones

Este proyecto buscaba ofrecer un servicio de intercambio de archivos como el que estamos acostumbrados a utilizar en La Nube, pero de una manera más segura y distribuida. El proyecto ha implementado una primera versión a una red de nodos superpuesta a la red física, que no depende de terceros para ninguna de sus actividades, permitiendo que los nodos intercambien archivos desde cualquier lugar de Internet. Cada uno de esos nodos podrá entrar y salir de la red sin que el desempeño del conjunto se vea afectado. Como prueba de concepto, se ha probado que esta idea es viable y todos los objetivos iniciales se han visto cumplidos.

La red se ha estructurado con el nodo en el núcleo. La estructura del nodo se ha diseñado e implementado de manera muy modularizada para que las diferentes partes sean independientes, aunque interactúen de manera eficiente. Se decidió hacerlo de esta manera para que los módulos pudieran ser sustituidos con facilidad en caso de que hubiera mejoras que hacer más adelante, sin afectar al desempeño del resto de la entidad del nodo. Este fue, de hecho, el caso cuando se implementó el primer cliente P2P (Ttorrent), pero después se decidió sustituirlo por otro (TomP2P), ya que el primero no cumplía las necesidades de descubrimiento de la red.

Desde el principio, la red buscó ser tan independiente como fuera posible de terceras partes, pero durante la implementación de Ttorrent en el módulo P2P, se hizo evidente que haría falta un tracker en un servidor ajeno para llevar a cabo las funciones de descubrimiento de la red. Al no estar esto en la línea del proyecto, Ttorrent se sustituyó por TomP2P, que ofrece un servicio de descubrimiento independiente de cualquier agente ajeno a la red. Este proceso se llama Bootstrapping y traerá consigo la necesidad de dos tipos de nodos diferentes: el nodo Master y el nodo normal, aunque la estructura del nodo se ha implementado de manera que todos ellos puedan jugar ambos papeles si fuera necesario. Sólo habrá un nodo Master de manera simultánea en la red y será el encargado de recibir las peticiones de entrada y llevar a cabo el proceso de Bootstrapping que le

dará al nodo entrante la información necesaria acerca de la topología de la red.

Aunque la conexión P2P entre los nodos ya dotaba a la red de una estructura y se pudieron intercambiar archivos entre los dispositivos, para conseguir las mismas funcionalidades que ofrecen estos servicios en La Nube, los archivos tenían que ser accesibles desde cualquier dispositivo con conexión a Internet. Análogamente a los servicios en La Nube, este problema se ha solucionado mediante un Interfaz Web accesible desde cualquier dispositivo que disponga de un navegador web. Así, el nodo incorpora una funcionalidad más, haciendo también las veces de servidor web: se encargará de servir la página web del interfaz y proveerá al usuario con las herramientas necesarias para gestionar los archivos de manera ubicua. Este interfaz será la ventana del usuario a la aplicación, conectándole con el nodo asociado a ese interfaz, así como permitiéndole tener el control de los archivos tanto en ese nodo como en el resto de la red.

Con respecto a la información intercambiada, habrá dos tipos de información viajando entre los nodos: mensajes con información acerca de los archivos disponibles, y los archivos en sí que se intercambien. Aunque ambos tipos de datos podrían haberse intercambiado a través del mismo módulo del nodo, se separaron de tal forma que el módulo HTTP se encargara de la información sobre los archivos, mientras que el módulo P2P intercambiara las propias partes de los archivos. Esta no fue una decisión trivial, ya que ofrecía dos ventajas: primero, la red P2P es mucho más eficiente a la hora de intercambiar grandes cantidades de datos, como ocurrirá cuando se envíen los archivos; segundo, para que el Interfaz Web sea capaz de comunicarse con los nodos de manera tan fluida como estos se comunican entre ellos, se ha implementado una API REST en el módulo HTTP. El Interfaz Web no es un nodo como los demás, por lo que es especialmente importante aunar las comunicaciones y eliminar la necesidad de estar traduciendo los mensajes de la web a mensajes que el nodo pueda entender.

En lo relativo a la gestión de los archivos, se dividirán en partes de tamaño fijo para ser compartidos. Esto hará que la descarga sea mucho más eficiente cuando se descargue de varios nodos a la vez. Al ser la velocidad de subida de los nodos emisores normalmente más baja que la velocidad de bajada del nodo receptor, si los datos son descargados de varios nodos de manera simultánea de varias fuentes, no ocurrirá un fenómeno de cuello de botella en los nodos emisores, pero se optimizará la velocidad de descarga del nodo receptor. La información sobre los archivos se gestionará mediante estructuras de datos diseñadas específicamente para este proyecto: una de ellas será un archivo XML que guardará la información cuando la aplicación se cierre, y la segunda será una estructura más accesible y dinámica durante la ejecución del programa.

En conclusión, los objetivos iniciales han sido satisfechos; las principales funcionalidades de un servicio de intercambio de archivos en La Nube se han implementado mediante una red Peer-to-Peer, complementándolo con una manera de acceder a los archivos desde cualquier punto de Internet. Sin embargo, si se hubiera dispuesto de más tiempo se podrían haber hecho mejoras. Más que solucionar errores, las mejoras estarían enfocadas a expandir la capacidad y las funcionalidades de la aplicación. Algunas ideas que podrían ser interesantes van desde convertir la aplicación en multi-dispositivo o mejorar la seguridad de las comunicaciones y del Interfaz Web.

C.2 Trabajo futuro

Durante el desarrollo de este proyecto se ha aprendido mucho, sobretodo acerca de redes P2P y gestión de archivos. Sin embargo, si se hubiera contado con más tiempo se habrían podido implementar mas funcionalidades que completaran la versión actual. Si el proyecto continuara, estos serían algunos de los aspectos que sería interesante incluir:

- Convertir la aplicación en multi-dispositivo:

  Como ya se explicó en la sección 4.2 Módulo P2P, TomP2P se seleccionó, entre otras razones, por disponer de una implementación en Android que permitiría que los nodos se ejecutaran también en dispositivos móviles y tablets. Al estar el resto de los nodos escritos en Java y ser bastante ligeros, la aplicación podría fácilmente migrarse a estos dispositivos.

  Si esto ocurriera, el módulo de políticas descrito en la sección 3.8 Módulo de Política sería necesario; hasta ahora no lo había sido dado que el desarrollo ha sido para dispositivos más grandes y sin restricciones. Aspectos como la batería de estos pequeños dispositivos o el hecho de que pudieran estar conectados a la red a través de datos móviles podría tener que tomarse en cuenta para no perjudicar al usuario cuando la aplicación se ejecute.

- Mejoras de seguridad:

  Especialmente en lo relacionado con el Interfaz Web, que es probablemente el punto más vulnerable de la red. Haría falta implementar métodos de autenticación con credenciales como el descrito en el diseño antes de que se publicara la aplicación. Aunque la experiencia general del usuario no se vería afectada por esta medida de entrada al Interfaz Web, sería una medida de seguridad importante.

  También sería importante tener un control de los nodos que acceden a la red a través del Bootstrapping con el nodo Master, que necesitaría comprobar que el nodo nuevo ha sido invitado por alguno de los nodos que ya conforman la red. Esto requeriría, sin

embargo, bastante tiempo para ser desarrollado y llevar a cabo más investigación acerca de este tipo de medidas de seguridad.

- Acceso más fácil a la red:

  Aunque la solución actual con una IP fija para el nodo Master es completamente funcional, funciona de manera eficiente en los casos testeados en los que había un número no demasiado alto de nodos. Sería recomendable estar preparados para un número mayor de nodos que compusieran la red, para lo que el servicio de DNS dinámico funcionaría mejor.

- Ofrecer más funcionalidades al usuario:

  Como primera versión de un servicio de intercambio de archivos, este proyecto cumple las necesidades básicas de un usuario, pero se podrían añadir más funcionalidades que completarían la experiencia. Estas nuevas opciones podrían ser, por ejemplo, la posibilidad de eliminar archivos de la red desde el Interfaz Web o la de hacer una copia de seguridad automática en el resto de los nodos de algunos archivos marcados como importantes por el usuario.

Bibliografía

1. X. Shen, H. Yu, J. Buford and M. Akon, Handbook of peer-to-peer networking. New York: Springer, 2010.

2. P. Maymounkov and P. Mazieres, Kademlia: A Peer-to-peer information system based on the XOR metric. First International Workshop on Peer-to-Peer Systems. 2002.

3. S. Srinivasan, Cloud computing basics. 2014.

4. B. Cohen, "The BitTorrent Protocol Specification", Bittorrent.org, 2013. [Online]. Available: http://www.bittorrent.org/beps/bep_0003.html

5. D. Bricklin, "Friend-to-Friend Networks", Bricklin.com, 2000. [Online]. Available: http://www.bricklin.com/f2f.htm

6. "Turtle F2F", www.wow.com, 2016. [Online]. Available: http://www.wow.com/wiki/Turtle_F2F

7. M. Petazzoni, "ttorrent", www.github.com, 2016. [Online]. Available: https://github.com/mpetazzoni/ttorrent

8. "TomP2P", Tomp2p.net, 2016. [Online]. Available: http://www.tomp2p.net

9. J. Clark and S. DeRose, "XML Path Language (XPath)", W3.org, 2015. [Online]. Available: https://www.w3.org/TR/xpath/

10. "Bootstrap", Getbootstrap.com, 2016. [Online]. Available: http://getbootstrap.com/

11. "jQuery", Jquery.com, 2016. [Online]. Available: http://jquery.com/

12. E. Van der Sar, "Filesharing Around The Globe - TorrentFreak", TorrentFreak, 2006. [Online]. Available: https://torrentfreak.com/filesharing-around-the-globe/

13. R. MUÑOZ, "PRIMERA SENTENCIA CIVIL QUE DECLARA LEGALES LAS REDES P2P DE DESCARGAS", EL PAÍS, 2010. [ONLINE]. AVAILABLE: http://tecnologia.elpais.com/tecnologia/2010/03/13/actualidad/1268474462_850215.html

14. "LAS REDES P2P SON LEGALES EN ESPAÑA: SENTENCIA A FAVOR DE PABLO SOTO", ELDIARIO.ES, 2014. [ONLINE]. AVAILABLE: http://www.eldiario.es/turing/propiedad_intelectual/Justicia-victoria-Pablo-Soto-discograficas_0_247775662.html

15. "CIRCULAR 8/2015, SOBRE LOS DELITOS CONTRA LA PROPIEDAD INTELECTUAL COMETIDOS A TRAVÉS DE LOS SERVICIOS DE LA SOCIEDAD DE LA INFORMACIÓN TRAS LA REFORMA OPERADA POR LEY ORGÁNIC A 1/2015", 2016. [ONLINE]. AVAILABLE: HTTPS://WWW.FISCAL.ES/FISCAL/PA_WEBAPP_SGNTJ_NFIS/DESCARGA/CIRCULAR_8-2015.PDF?IDFILE=D7B418C1-6AA4-406A-BCEB-19B2A9495C52